UNIVERSIDADE FEDERAL DO RIO GRANDE

PAULO SANTOS NETO

2T-FT: Fine-Tuning com Apenas Dois Tokens para Aprimorar o Desempenho de LLMs Universidade Federal do Rio Grande

Brasil

2025

UNIVERSIDADE FEDERAL DO RIO GRANDE

PAULO SANTOS NETO

2T-FT: Fine-Tuning com Apenas Dois Tokens para Aprimorar o Desempenho de LLMs Universidade Federal do Rio Grande

Trabalho acadêmico apresentado ao Curso de Engenharia de Computação da Universidade Federal do Rio Grande, como requisito parcial para a obtenção do grau de Bacharel em Engenharia de Computação

Orientador: Rodrigo da Silva Guerra

Coorientador: Felipe Kühne

Universidade Federal do Rio Grande – FURG

Centro de Ciências Computacionais

Curso de Engenharia de Computação

Brasil

2025

Agradecimentos

Gostaria de expressar minha sincera gratidão ao meu orientador, Professor Rodrigo Guerra, pela orientação, apoio e expertise ao longo deste trabalho. Agradeço também ao meu coorientador, Felipe Kühne, pela valiosa contribuição e incentivo. O conhecimento e as orientações de ambos foram fundamentais para a realização deste projeto.

Resumo

Os grandes modelos de linguagem (*Large Language Models*, LLMs) se estabeleceram como o estado da arte no processamento natural de linguagem (*Natural Language Processing*, NLP), expandindo seu alcance nas mais diversas áreas de inteligência artificial. Entre as variadas categorias de tarefas que tais modelos são capazes de lidar, destaca-se a capacidade de "raciocínio" durante a generação de texto — isto é, a habilidade de resolver problemas que demandam a construção de uma sequência lógica de pensamentos (*Chain-of-Thought*, CoT) que conduzam a uma solução coerente. O método de *CoT* tem demonstrado melhorias no desempenho de *LLMs* em uma ampla variedade de tarefas, incluindo raciocínio aritmético, de senso comum e simbólico. No entanto, estas melhorias tipicamente exigem o desenvolvimento de *prompts* CoT eficazes. Por outro lado, trabalhos mais recentes mostram que os caminhos de raciocínio CoT já estão frequentemente presentes de forma inerente nas k melhores sequências alternativas de decodificação, mesmo na ausência de uma técnica de *prompting* específica. Neste trabalho, propõe-se um novo método de *fine-tuning* que explora essa propriedade ao focar em apenas dois *tokens* específicos dessas respostas CoT preexistentes. Demonstra-se que é possível incrementar significativamente a geração de respostas contendo CoT em uma decodificação gulosa ao realizar o *fine-tuning* de apenas dois *tokens*, utilizando os CoTs implícitos gerados pelo modelo. Essa abordagem permite que os LLMs sejam mais eficientes ao lidar com problemas de raciocínio, ao mesmo tempo em que se reduz o custo computacional associado à sua implementação em cenários práticos.

Palavras-chave: LLM. Chain-of-Thought. LoRA. Fine-tuning. Reasoning.

Abstract

Large Language Models (LLMs) have established themselves as the state of the art in natural language processing (NLP), expanding its reach in the most diverse areas of artificial intelligence. Among the various categories of tasks these models can handle, their capacity for reasoning during text generation stands out—namely, the ability to solve problems that require constructing a logical sequence of thoughts (chain-of-thoughts) leading to a coherent solution. The chain-of-thought (CoT) method has shown improvements in LLM performance across a wide range of tasks, including arithmetic, commonsense, and symbolic reasoning. However, these improvements typically require the development of effective CoT prompts. On the other hand, more recent work shows that CoT reasoning paths are often inherently present in the alternative top-k decoding sequences, even in the absence of specific prompting techniques. In this paper, we propose a novel fine-tuning method that leverages this property by focusing on just two specific tokens from these pre-existing CoT responses. We demonstrate that it is possible to significantly enhance the generation of CoT-containing responses in greedy decoding by fine-tuning only two tokens, utilizing the implicit CoTs generated by the model. This approach allows LLMs to be more efficient in solving reasoning tasks while reducing the computational cost associated with their implementation in practical scenarios.

Keywords: LLM. Chain-of-Thought. LoRA. Fine-tuning. Reasoning.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

LoRA	Low-rank Adaptation
LLM	Large Language Model
CoT	Chain-of-Thought
BERT	Bidirectional Encoder Representations from Transformers
PEFT	Parameter-Efficient Fine-Tuning
GPT	Generative Pre-trained Transformer
SOTA	State-of-the-Art
NLP	Natural Language Processing
ICL	In-context Learning
LeCo	Learning From Correctness
RoPE	Rotary Positional Embedding
EOS	End-of-Sequence
GPU	Graphics Processing Unit

Sumário

1 Introdução

Nos últimos anos, os modelos de linguagem de larga escala (LLMs) se estabeleceram como o estado da arte no processamento de linguagem natural (NLP), expandindo seu alcance para diversas áreas dentro da Inteligência Artificial (XIE et al., 2022; TOUVRON et al., 2023; JIANG et al., 2023; BROWN et al., 2020a). Entre as múltiplas tarefas que esses modelos conseguem realizar, destaca-se a capacidade de "raciocínio", isto é, a habilidade de solucionar problemas que requerem não apenas a recuperação de informações específicas armazenadas nas camadas internas do modelo, mas também a construção de uma sequência lógica de argumentos que conduza a uma solução coesa (HUANG; CHANG, 2023; QIAO et al., 2023). A constituição destas múltiplas etapas para solução de um problema em *LLMs* é conhecido como cadeia de pensamentos (ou *CoT*) (WEI et al., 2023; WANG et al., 2023; NYE et al., 2021), e tem se tornado cada vez mais proeminente devido à sua capacidade de induzir raciocínio em tais modelos. Apesar da crescente quantidade de pesquisa acadêmica dedicada a este escopo, a resolução de problemas deste gênero continua sendo um desafio em aberto, principalmente para questões que necessitam maior complexidade de raciocínio para sua solução (FU et al., 2023).

Nesta perspectiva, uma das técnicas comumente empregadas para elicitar o raciocínio nestes modelos é através do ajuste fino (*fine-tuning*) de conjuntos de dados projetados para estimular essa habilidade. Tal técnica é conhecida como ajuste fino por instrução (ou *instruction fine-tuning*) (WEI et al., 2022b; CHUNG et al., 2022), e refere-se à união do *supervised fine-tuning* com adição de caminhos *CoT* para o treinamento mais robusto de modelos de linguagem, onde o modelo é treinado usando pares de instruções de entrada e suas respectivas saídas desejadas. Este método permite que o modelo aprenda tarefas específicas guiadas por essas instruções, melhorando sua capacidade de compreender e executar tarefas com base nas instruções fornecidas. Concomitante a isto, não somente o *prompt* é formato como um conjunto de instruções, mas também apresenta em sua composição uma quantidade representativa de dados *Chain-of-Thoughts* (CoT) (COBBE et al., 2021). Contudo, tal técnica apresenta suas limitações, principalmente quanto a dependência de disponibilidade e qualidade dos conjuntos de dados de treinamento, além de exigir recursos computacionais significativos.

Em uma outra óptica, há um foco crescente na técnica de *prompt engineering* como uma alternativa ao *fine-tuning* tradicional (CHEN et al., 2023a; SAHOO et al., 2024). O *prompt engineering* envolve projetar entradas específicas para o modelo, conhecidas como *prompts*, que orientam sua resposta para uma tarefa desejada. Essa abordagem pode incluir técnicas como *zero-shot* e *few-shot learning*, permitindo que o modelo responda a tarefas para as quais não foi explicitamente treinado (ZHOU et al., 2023; CHEN et al.,

2023b; GAO et al., 2023a). Uma das principais vantagens do *prompt engineering* é sua capacidade de economizar recursos computacionais, uma vez que não é necessário retreinar o modelo. No entanto, o *prompt engineering* também apresenta desafios. A formulação adequada dos *prompts* pode exigir um entendimento profundo da tarefa e do modelo, bem como um maior custo computacional durante o tempo de inferência.

Em contraste com esses métodos, Wang e Zhou (2024a) identifica uma nova óptica sobre a composição de *Chain-of-Thoughts* intrísinca aos modelos: as cadeias de pensamento podem ser derivadas exclusivamente pela exploração no processo de decodificação do modelo durante a autoregressão, sem a necessidade de treinamento por *instruction tuning* ou uso de *prompt engineering*. Além disso, a extração desses caminhos CoT revela que o modelo tende a ter uma confiabilidade maior em relação à resposta final, quando comparado com a decodificação gananciosa (*greedy decoding*), que geralmente resulta em respostas mais curtas e diretas. Esse processo será melhor explicado na seção 2.6.

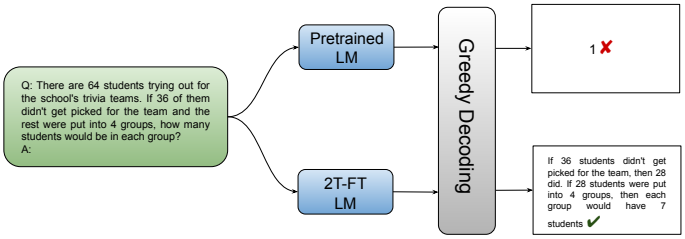
Neste trabalho se adotou a metodologia exemplificada por Wang e Zhou (2024b). Por meio da análise das primeiras K respostas geradas pelo modelo, extrai-se um subconjunto composto por respostas com maior presença de características associadas ao CoT. Este subconjunto representa os caminhos/respostas CoT implícitos ao modelo, considerando que não foram previamente induzidos por qualquer técnica de *prompting*. Esta técnica é denominada *CoT-Decoding*.

Sob essa perspectiva, e alinhando-se à ideia de que modelos de linguagem ajustados finamente podem melhorar o desempenho em cenários *Zero-Shot* (WEI et al., 2022a), este trabalho propõe uma abordagem de *fine-tuning* altamente econômica, implementada por meio do treinamento de apenas dois *tokens* dos caminhos CoT implícitos ($2T$-FT). Em específico, dada uma janela de contexto de 512 *tokens*, que compreende a combinação da pergunta, resposta e *tokens* especiais (EOS/PAD), apenas dois *tokens* são treinados: o primeiro *token* da resposta e o primeiro *token* de *end-of-sentence* (EOS).

Como ilustrado na Figura 1, o objetivo é promover a geração de CoT durante o processo de *decoding*, no qual são selecionados apenas os *tokens* mais prováveis (*greedy decoding*), aumentando, assim, a probabilidade de o modelo produzir respostas corretas.

Ao aprimorar a geração de raciocínio nos estágios iniciais do processo de decodificação após o ajuste fino do modelo pré-treinado com dois *tokens*, mantém-se o custo computacional de geração inalterado, sem a necessidade de explorar múltiplos caminhos K durante o tempo de inferência. Ademais, ao priorizar o *backpropagation* em apenas dois *tokens* da resposta durante o *fine-tuning*, evidencia-se uma redução drástica no poder computacional necessário para o *fine-tuning* quando comparado com outros métodos. Além disto, o método alinham-se a estudos recentes que demonstram a eficácia da seleção de *tokens* específicos no processo de treinamento (LIN et al., 2024).

Figura 1 – Comparação de saídas entre o modelo pré-treinado e o 2T-FT.



Fonte: Próprio Autor.

1.1 Objetivos Gerais e Específicos

O objetivo geral deste trabalho é introduzir uma nova técnica de *fine-tuning*, fundamentada no uso de *CoTs* implícitos aos modelos de linguagem, bem como no uso de apenas dois *tokens* durante o *fine-tuning* supervisionado do modelo. Deste modo, os objetivos são:

- **Investigar a eficiência de ajustes mínimos em *tokens* específicos durante o processo de *fine-tuning***. Especificamente, busca-se avaliar como o treinamento de apenas dois *tokens* — o primeiro da resposta e o primeiro *EOS* — pode impactar a capacidade do modelo em gerar respostas corretas e coerentes.
- **Desenvolver um método para identificar caminhos implícitos contendo *CoT* com o objetivo de automatizar o processo de *fine-tuning***. Isto implica na automatização da etapa intermediária capaz de selecionar corretamente os *tokens* que definem o escopo da resposta final do modelo.
- **Superar limitações associadas às abordagens tradicionais de *instruct fine-tuning* e *prompt engineering***. Propor uma solução alternativa e eficiente para guiar modelos de linguagem, com foco em evidenciar e ampliar sua capacidade de raciocínio em cadeia (*chain-of-thoughts*) sem o uso de técnicas de *prompt*.

A partir dos pontos supracitados, visa-se superar as limitações e desafios associados ao *instruct fine-tuning* e ao *prompt engineering*, oferecendo uma abordagem alternativa e eficiente para direcionar modelos de linguagem, com potencial para evidenciar a capacidade de generação de *chain-of-thoughts* dos modelos.

1.2 Estrutura do Documento

No Capítulo 2, é realizada uma breve exploração da área de Inteligência Artificial (IA) e redes neurais, com ênfase na arquitetura das redes *Transformers* e sua relevância no

campo da IA. Este capítulo também inclui uma discussão aprofundada sobre os grandes modelos de linguagem e suas especificidades, além de uma contextualização sobre os avanços recentes no setor, abordando técnicas aplicadas para ajuste fino em Modelos de Linguagem de Grande Escala, indução de comportamento através da Engenharia de *Prompt* (*Prompt Engineering*), geração de *CoT*, e os diversos métodos de decodificação utilizados durante o processo de autoregressão do modelo.

No Capítulo 3, são envidenciadas as metodologias empregadas, as quais são divididas em duas etapas: generalização do método de *CoT-Decoding* usando modelo *BERT* e formalização do método *2T-FT*.

O Capítulo 4 apresenta a validação da técnica por meio de experimentos qualitativos e quantitativos usando diferentes combinações de técnicas. Além disso, é realizada uma comparação entre o desempenho de modelos distintos, juntamente com uma análise do tempo de inferência do modelo após o ajuste fino e a remoção do *CoT-Decoding*.

Por fim, o Capítulo 5 aborda o cronograma de atividades durante o período da elaboração da tese.

2 Fundamentação Teórica

Esta seção apresenta os conceitos fundamentais que servem de base para o desenvolvimento do trabalho. Inicialmente, aborda-se a Inteligência Artificial e Redes Neurais, destacando sua evolução e relevância no contexto atual. Em seguida, são discutidas as redes *Transformers*, que caracterizam a infraestrutura base dos grandes modelos, e os *LLMs*, que compõem os principais modelos de linguagem da atualidade.

Na sequência, exploram-se as Técnicas para *Fine-tuning*, utilizadas para adaptar modelos pré-treinados a tarefas específicas. Também é discutido o Raciocínio em *LLMs*, com foco na capacidade desses modelos de realizar inferências complexas. Por fim, apresentam-se os Métodos de Decodificação, responsáveis pela forma como é dada a geração textual do modelo.

2.1 Inteligência Artificial e Redes Neurais

A Inteligência Artificial (IA) representa um campo interdisciplinar responsável pela elaboração de sistemas capazes de simular e automatizar tarefas cognitivas por meio de recursos computacionais e formulações matemáticas. Inicialmente centrada em modelos mais específicos para determinadas aplicações, como máquinas de vetores de suporte (SVM), árvores de decisão, algoritmos genéticos e sistemas baseados em lógica difusa, a inteligência artificial se disseminou como um conjunto de ferramentas capazes de solucionar problemas através de aprendizado computacional. Nesse contexto, as redes neurais artificiais surgiram como uma ferramenta matemática fundamental para aprender a partir de dados e lidar com tarefas complexas de maneira mais genérica. Uma rede neural artificial é um modelo computacional inspirado na estrutura dos neurônios do cérebro humano, composta por camadas de unidades interconectadas, cada uma realizando operações simples. Tais redes têm sido amplamente utilizadas em problemas de reconhecimento de padrões, processamento de linguagem natural, visão computacional e diversas outras áreas da inteligência artificial (RUSSELL; NORVIG, 2010).

Redes neurais artificiais são especialmente eficazes em lidar com problemas complexos e não lineares, que são desafiadores para métodos tradicionais de programação (KUPTSOV; KUPTSOVA; STANKEVICH, 2021). Através de um processo de treinamento supervisionado ou não supervisionado, essas redes são capazes de ajustar seus parâmetros internos para otimizar o desempenho em uma determinada tarefa. O surgimento de algoritmos de aprendizado de máquina (*machine learning*) e aprendizado profundo (*deep learning*) impulsionou significativamente o avanço das redes neurais, permitindo o processamento de grandes volumes de dados e a resolução de problemas anteriormente

considerados intratáveis (GOODFELLOW; BENGIO; COURVILLE, 2016).

Dentre as diversas arquiteturas formuladas a partir dos conceitos fundamentais em redes neurais, destacam-se as Redes Neurais Convolucionais (CNNs) que são amplamente utilizadas em tarefas de visão computacional, devido à sua capacidade de capturar padrões espaciais em dados de entrada através de camadas convolucionais e de pooling (LECUN et al., 1998). Para tarefas envolvendo sequências temporais, como processamento de linguagem natural, destacam-se as Redes Neurais Recorrentes (RNNs) e suas variantes, como *Long Short-Term Memory* (LSTM), que lidam bem com a dependência temporal ao processar sequências de dados (HOCHREITER; SCHMIDHUBER, 1997). Recentemente, as Redes *Transformers* emergiram como uma arquitetura revolucionária para o processamento de sequências sem necessitar de recorrências, com aplicação em tradução automática, sumarização de texto e modelagem de linguagem (VASWANI et al., 2017). Essa arquitetura se destaca por sua capacidade de aprender relações globais em conjuntos de dados sequenciais, utilizando mecanismos de atenção.

2.2 Redes Transformers

As Redes Transformers representam uma classe poderosa de modelos de aprendizado profundo que revolucionaram o processamento de linguagem natural (NLP) e outras tarefas sequenciais. Introduzido por Vaswani et al. (2017), o modelo *Transformer* é construído em torno de mecanismos de atenção que permitem que o modelo direcione e pondere diferentes partes da entrada durante o processamento. A arquitetura original do *Transformer* baseia-se em uma estrutura de *encoder-decoder*, na qual são realizadas operações de atenção e transformações não lineares, substituindo modelos anteriores, como redes recorrentes ou convolucionais.

Antes de ser integrado à arquitetura *encoder-decoder*, algumas etapas são necessárias para transformar o texto bruto em dados compreensíveis para o modelo. Primeiramente, o texto fornecido pelo usuário (ou *prompt*), no formato *string*, é verificado quanto ao seu comprimento em relação à janela de contexto definida pelo modelo. A janela de contexto delimita o número máximo de *tokens* que podem ser processados em uma única inserção.

Caso o número de *tokens* seja inferior ao limite máximo, os espaços restantes são preenchidos com *tokens* especiais, como *end-of-sentence* ou *end-of-sequence* (*EOS*). Por outro lado, se o número de *tokens* exceder a capacidade da janela de contexto, o texto precisa ser subdividido em blocos (*chunks*), que são processados sequencialmente pelo modelo em múltiplas etapas. Esse pré-processamento garante que o texto seja formatado de maneira adequada para utilização pela arquitetura *Transformer*.

A etapa inicial consiste na transformação do texto, representado como uma *string*, em um *embedding*, ou seja, em valores numéricos organizados em um sistema matemático

de N dimensões. Esse processo é realizado pelos componentes de *input embedding* ou *output embedding* (Figura 2). Como resultado, cada *token* do texto é representado por um vetor em D dimensões, o qual codifica, de forma superficial, relações semânticas entre os *tokens*.

Em seguida, com o objetivo de atribuir uma estrutura posicional aos *tokens*, realiza-se o processo de *positional encoding*. Esse procedimento adiciona pequenos valores específicos a cada vetor, permitindo que a rede identifique a ordem dos *tokens* sem depender de uma estrutura explicitamente sequencial, como ocorre nas redes recorrentes (*RNNs*). Essa abordagem capacita o modelo a processar informações de maneira paralela e eficiente, característica fundamental das arquiteturas baseadas em *Transformers*.

Após as transformações realizadas no texto, este é inserido na arquitetura base do *Transformer*. O funcionamento dessa arquitetura pode ser explicado da seguinte maneira: o modelo é dividido em duas partes principais, o codificador (*encoder*) e o decodificador (*decoder*). O codificador tem a função de processar a sequência de entrada e convertê-la em uma representação interna. Por exemplo, ao receber a frase "Eu gosto de livros", o codificador utiliza camadas de atenção *multi-head* e redes *feedforward* para identificar as relações entre as palavras da sequência, como a conexão semântica existente entre "gosto" e "livros".

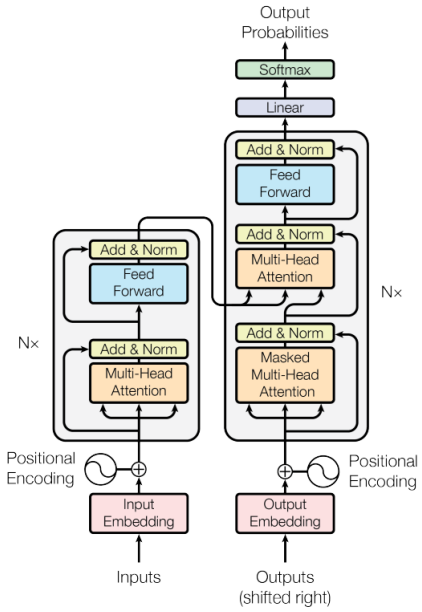
O decodificador, por sua vez, utiliza essa representação interna gerada pelo codificador para produzir a sequência de saída, palavra por palavra. No caso de uma tradução para o inglês, o decodificador iniciaria com a palavra "I" e, utilizando as informações fornecidas pelo codificador, preveria a próxima palavra "like", e assim sucessivamente, até formar a frase completa "I like books". Durante esse processo, os *tokens* previamente gerados são incorporados como contexto, garantindo a consistência e a coerência da saída.

De forma mais aprofundada, o elemento central que diferencia o *Transformer* de outras arquiteturas de inteligência artificial é o mecanismo de atenção. Esse componente fundamental permite ao modelo identificar e atribuir pesos às partes mais relevantes ou importantes dos dados em um determinado contexto, garantindo que a relação entre elementos da sequência seja processada de maneira eficaz e adaptativa. Subdivido em três matrizes de pesos denominadas *query* (Q), *key* (K) e *value* (V), e estruturada em múltiplas cabeças de atenção, o mecanismo possibilita a captura de relações entre pares de sequências em múltiplos contextos. Isto é feito pelo produto escalar entre as camadas de *query* e *key*, escalonados pela dimensionalidade dos dados (d_k) e normalizados por uma função *softmax*. Ao final, aplica-se novamente um produto escalar entre o resultado anterior e a matrizes de pesos *value*, como na Equação 2.1.

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.1}$$

Por meio de uma perspectiva global apresentada na ilustração da Figura 2, podem-se observar as duas categorias de mecanismos de atenção que constituem o modelo originalmente proposto *Transformer*: *Multi-Head Attention* e *Masked Multi-Head Attention*. O primeiro encontra-se na camada *encoder* e possibilita uma relação de auto-atenção (*self-attention*) entre os dados de entrada do modelo. O segundo está presente na camada *decoder* e é fundamental para impedir que o modelo tenha acesso às sequências de *tokens* a serem preditas. Uma camada *feed forward* é adicionada ao final, aumentando significativamente o número de parâmetros e, consequentemente, a representatividade do modelo. Por fim, as conexões residuais e a normalização em camadas constituem as saídas dos mecanismos de atenção e das camadas *feed forward*, com o intuito de mitigar o problema da degradação do gradiente e aprimorar a robustez durante o treinamento, respectivamente (BA; KIROS; HINTON, 2016; HE et al., 2015).

Figura 2 – Arquitetura Transformer



Fonte: Vaswani et al. (2017, p. 3)

Arquitetura de uma Rede *Transformer Vanilla*, formada por N blocos *encoder-decoder*

O uso da arquitetura *Transformer* no processo de treinamento e inferência segue etapas distintas, mas complementares, que garantem a adaptação do modelo e sua capacidade de generalizar para novas entradas.

Durante o treinamento, o modelo é alimentado com pares de entrada e saída, como no caso de tarefas de tradução, onde o modelo aprende a mapear uma sequência de palavras em uma língua para uma sequência equivalente em outra. O objetivo é ajustar os parâmetros da rede para minimizar a discrepância entre a saída gerada pelo modelo e a

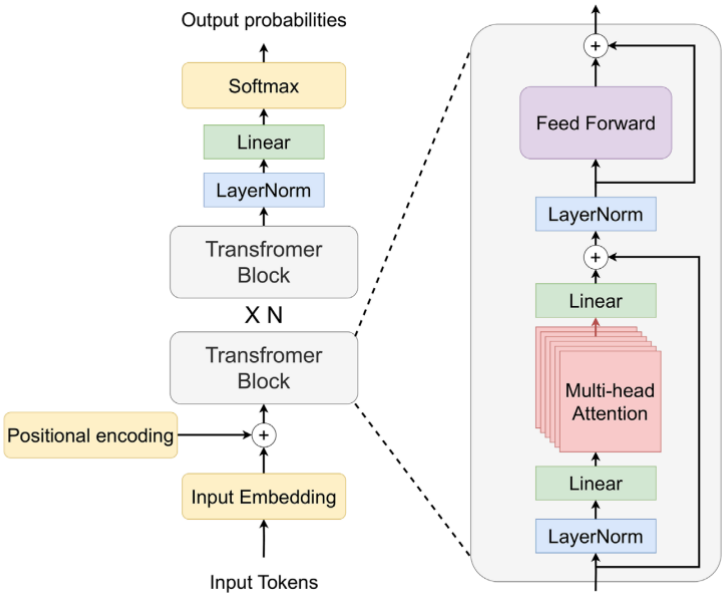
saída real, utilizando funções de perda apropriadas e algoritmos de otimização, como o *Adam* (KINGMA; BA, 2017). Nesse processo, o mecanismo de atenção desempenha um papel crucial, pois permite que o modelo identifique e atribua diferentes pesos às partes da entrada que são mais relevantes para a geração de cada *token* da sequência de saída, promovendo um aprendizado mais eficiente das relações contextuais.

Na fase de inferência, o modelo, já treinado, é utilizado para fazer previsões em novas entradas. Por exemplo, ao ser fornecida uma frase para tradução, o codificador processa a sequência de entrada e gera uma representação interna, que é utilizada pelo decodificador para gerar a sequência de saída, *token* por *token*. Durante a inferência, o modelo utiliza a informação adquirida no treinamento, incluindo os pesos de atenção, para prever os *tokens* mais prováveis em cada etapa. Técnicas de decodificação, como *greedy search* ou *beam search*, são empregadas para escolher a sequência mais adequada de *tokens*, com o objetivo de gerar uma resposta precisa e coerente.

Desde sua introdução, os *Transformers* evoluíram em várias arquiteturas aprimoradas, como o *BERT* (DEVLIN et al., 2019) e o *GPT* (RADFORD et al., 2018), que alcançaram resultados de estado da arte em uma variedade de tarefas de *NLP*, como classificação de texto, resposta a perguntas e tradução. Além disso, outras modificações foram feitas na estrutura original do *Transformer*, modificando a ordem da camadas de normalização (XIONG et al., 2020), variações nas cabeças de atenção (SHAZEER, 2019; AINSLIE et al., 2023), e modificações nos *embeddings* de posição (SU et al., 2023). Estes modelos demonstram uma capacidade excepcional de generalização, aprendendo representações ricas e universais de linguagem a partir de grandes quantidades de dados não supervisionados. Este sucesso evidencia o potencial dos *Transformers* não apenas em *NLP*, mas também em outras áreas como visão computacional e biologia computacional, onde o processamento de sequências é essencial (DOSOVITSKIY et al., 2021; YUN et al., 2020; VERMA; BERGER, 2021).

Além das variações mencionadas anteriormente, os métodos recentes são predominantemente estruturados apenas com camadas *decoder* (Figura 3). O uso exclusivo de camadas *decoder* demonstrou ser suficientemente robusto para gerar sequências, mesmo quando emprega apenas mecanismos causais nos blocos de atenção. Além disso, tais mecanismos oferecem vantagens notáveis, principalmente quando avaliados em termos de capacidade de generalização em *zero-shot* (WANG et al., 2022), redução do tamanho do modelo, compartilhamento de parâmetros, otimização ideal para técnicas como *KV cache* (POPE et al., 2022) e, especialmente, treinamento com dados não anotados. Essas arquiteturas utilizam atenção autorregressiva, que mantém a classificação completa e, teoricamente, oferece uma capacidade expressiva superior em comparação com arquiteturas como *encoder-decoder* (DONG; CORDONNIER; LOUKAS, 2021).

Figura 3 – Arquitetura Transformer apenas com camadas decoder (*decoder-only*).



Fonte: Chalvatzaki et al. (2023, p. 7)

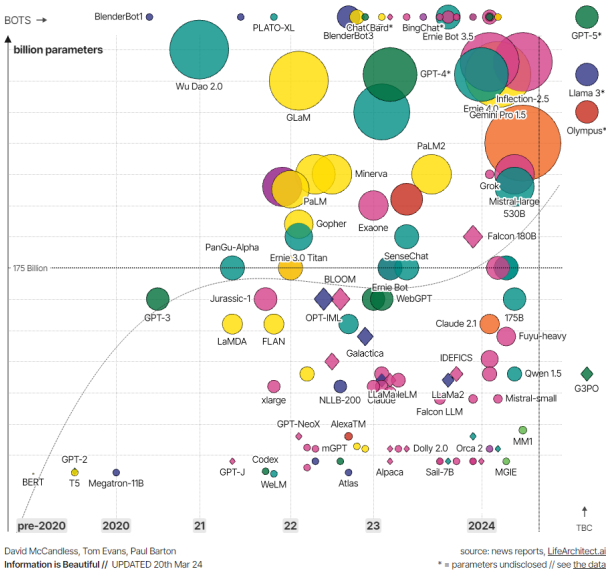
2.3 Large Language Models

Os modelos de linguagem de larga escala têm experimentado um crescimento exponencial nos últimos anos, impulsionado sobretudo pelo surgimento da arquitetura *Transformer*. Inicialmente quantificados na ordem dos milhares de parâmetros, esses modelos agora atingem a escala dos bilhões e até mesmo trilhões de parâmetros, conforme ilustrado na Figura 4. Embora o custo computacional associado a um grande número de parâmetros seja significativo, esse cenário proporciona benefícios notáveis, como uma capacidade aprimorada de generalização para tarefas subsequentes (*downstream tasks*), bem como uma representatividade mais abrangente das amostras (KAPLAN et al., 2020). Entre os principais modelos, estão o GPT (BROWN et al., 2020b), GEMINI (TEAM et al., 2024), CLAUDE 3 (ANTHROPIC, 2024) e PALM-2 (ANIL et al., 2023).

Os modelos supracitados são também caracterizados inicialmente como modelos pré-treinados (ou *pre-trained language models, PLMs*). Este pré-treinamento é realizado de forma auto-supervisionada em um grande corpus de texto, que pode chegar na casa dos trilhões de *tokens*, com alto custo computacional e financeiro. O principal objetivo desse processo é a extração de características fundamentais da língua, como o reconhecimento de estruturas gramaticais e a compreensão de semântica contextual (PETERS et al., 2018; LEWIS et al., 2019).

Em uma outra perspectiva, com o propósito de tornar estes modelos mais acessíveis, têm sido propostas novas arquiteturas com o objetivo de reduzir o número de parâmetros

Figura 4 – Dimensionalidade dos LLMs



Fonte: Information is Beautiful (2024)

Comparativo durante os anos entre os número de parâmetros usados para cada LLM.

sem comprometer o estado da arte (*State-of-the-Art*, *SOTA*) em desempenho. Destaca-se o modelo *Mixtral* (JIANG et al., 2024), que se concentra na composição de redes especialistas (*experts*), e o *Phi* (GUNASEKAR et al., 2023a), caracterizado pelo baixo número de parâmetros aliado ao treinamento em conjuntos de dados de qualidade excepcional, alcançando desempenho equivalente a modelos 25 vezes maiores do que ele próprio. Alguns outros modelos seguem esta abordagem de redução de parâmetros mantendo alto desempenho (SANH et al., 2020; TIAN et al., 2024; MITRA et al., 2023).

Nesta perspectiva, muitas outras técnicas são aplicadas em Modelos de Linguagem de Grande Escala com o intuito de acelerar o treinamento e a inferência, reduzir a complexidade computacional, aprimorar a eficiência de memória e otimizar o consumo de *GPU*. Alguns métodos focam na redução da precisão dos números flutuantes que armazenam os valores dos pesos (ou parâmetros) do modelo, processo denominado quantização (ZHAO et al., 2024; MA et al., 2024). Alterações na arquitetura dos mecanismos de atenção também são realizadas com o objetivo de reduzir a complexidade quadrática do mecanismo (DING et al., 2023) ou tornar mais eficiente o acesso às *GPUs* (DAO, 2023).

Em uma outra perspectiva, e visando solucionar problemáticas relacionadas a dados enviesados, toxicidade e confiabilidade nos dados fornecidos pelas *LLMs*, técnicas como *Reinforcement Learning from Human Feedback* (*RLHF*) têm se tornado proeminentes em tais modelos. Através destes métodos, é possível que o modelo atualize seus parâmetros por meio do *feedback* humano, alinhando-o a respostas mais coerentes e preferíveis aos

usuários (*alignment tuning*). O *InstructGPT* (OUYANG et al., 2022) é um exemplo de modelo adaptado para suportar *RLHF*.

Por fim, as *LLMs* demonstraram uma grande capacidade de interação com outras ferramentas e, principalmente, de atualizar suas respostas em tempo real, através do uso de *retrieval augmentation generation* (*RAG*) (LEWIS et al., 2021). Além das técnicas de *fine-tuning*, o modelo pode ser aprimorado por meio da unificação de técnicas de *prompt engineering* com a coleta de dados de fontes externas. Isso permite que seja embutido no *prompt* um contexto que o modelo não possuía previamente no treinamento ou *fine-tuning*. O *RAG* tem se mostrado benéfico não apenas na projeção de novas informações ao modelo, mas também em mecanismos de auto-correção e *reasoning* (MELZ, 2023).

2.4 Técnicas para Fine-tuning

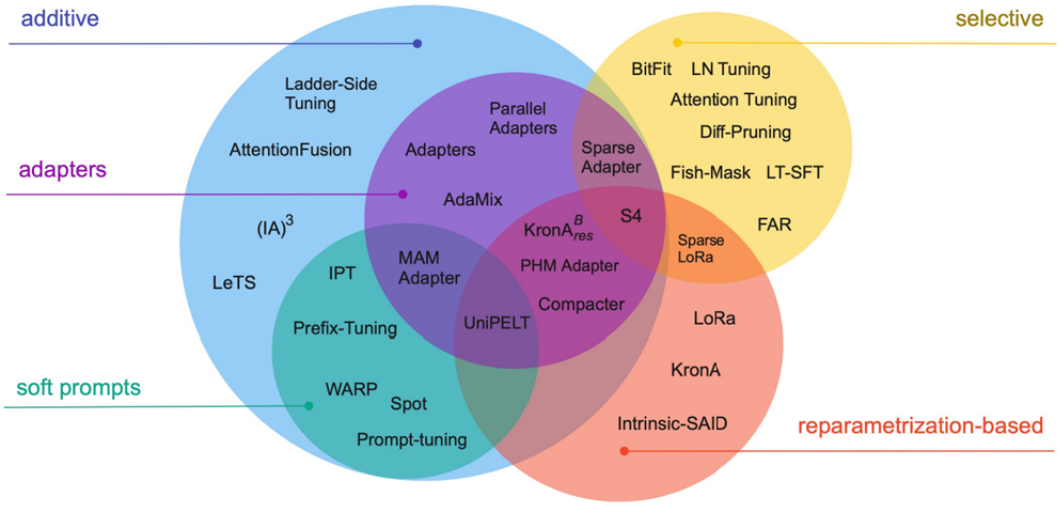
O processo de *fine-tuning* em *LLMs* refere-se à adaptação de um modelo pré-treinado a uma tarefa específica (*downstream task*) ou a um conjunto de dados particular. Esse processo é essencial para personalizar o desempenho do modelo e melhorar sua eficácia em contextos específicos. A título de exemplificação, a transformação do modelo pré-treinado para gerar texto em um modelo ajustado para responder perguntas é um processo de *fine-tuning* amplamente efetuado (YANG; YIH; MEEK, 2015; RAJPURKAR et al., 2016).

Tratando-se de modelos menores, na casa das centenas de milhões de parâmetros, o processo de *fine-tuning* pode ser realizado em todas as suas matrizes de pesos sem um custo computacional tão elevado. Entretanto, com o advento de modelos maiores, na casa das dezenas e centenas de bilhões de parâmetros, torna-se inviável ajustar todos os pesos do modelo sem a disponibilidade de grandes *data centers* com dezenas ou até centenas de *GPUs*. Por este motivo, muitas soluções foram desenvolvidas com o objetivo de minimizar o número de parâmetros necessários para a especialização do modelo em uma determinada tarefa, mantendo ao mesmo tempo a performance e a acurácia próximas ao estado da arte.

O *PEFT* (*Parameter-Efficient Fine-Tuning*) (XU et al., 2023a), é notavelmente um dos grandes avanços que engloba os métodos de *fine-tuning* que visam o uso de apenas uma parte dos pesos para o ajuste. Como ilustrado na Figura 5, os recentes métodos de ajuste fino podem ser subdivididos em cinco gêneros: *additive fine-tuning*, *partial/soft fine-tuning*, *reparemeterized fine-tuning*, *hybrid/selective fine-tuning* e *unified/adapters fine-tuning*.

Em síntese, o *additive fine-tuning* abrange os métodos nos quais há adição de novos pesos por meio de novas camadas ou modificações nos *embeddings* da rede. O *partial fine-tuning* foca na redução do número de parâmetros para o ajuste fino por meio de técnicas de *pruning*. O *reparameterized fine-tuning* não incrementa pesos adicionais, mas apenas modifica os pesos originais a partir da composição dos pesos pré-treinados com

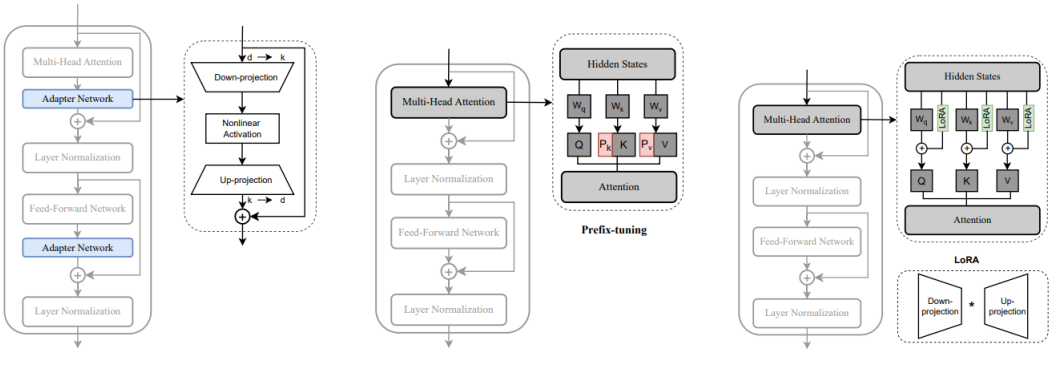
Figura 5 – Métodos de Fine-tuning.



Fonte: Lialin et al. (2024, p. 8)

pesos ajustados em matrizes substancialmente menores que as originais. Por fim, o *hybrid fine-tuning* e o *unified fine-tuning* representam um conjunto de técnicas que utilizam diversos outros métodos *PEFT* com o objetivo de tornar o ajuste mais robusto. A título de elucidação, a Figura 6 apresenta um comparativo entre as arquiteturas dos três métodos anteriormente citados.

Figura 6 – Detalhes de Arquitetura em *Sequential Adapter*, *Prefix-Tuning* e *LoRA* .



(a) Sequential Adapter (b) Prefix-tuning (c) LoRA

Fonte: Xu et al. (2023a, p. 5)

Os métodos de reparametrização são amplamente explorados, destacando-se principalmente o *LoRA* (*Low-rank Adaptation*) (HU et al., 2021) e suas derivações (VALIPOUR et al., 2023; HUANG et al., 2024a; LIU et al., 2024). Em síntese, o *LoRA* permite a reparametrização dos pesos originais do modelo por meio do treinamento de duas matrizes

significativamente menores do que as camadas lineares do modelo original. Com apenas uma fração ínfima do número de pesos em comparação ao tamanho original do modelo, o *LoRA* possibilita que o *fine-tuning* alcance resultados aproximadamente similares ao *fine-tuning* completo, utilizando muito menos poder computacional e tempo de treinamento. Além disso, por ser um método de reparametrização, é possível adicionar ou remover as matrizes *LoRA* de maneira simples e rápida, tornando-o um método mais versátil e dinâmico. Embora inicialmente validado em tarefas de linguagem natural, o LoRA também possui aplicações em geração de imagens, como na estilização de imagens (WANG et al., 2023).

Concomitante ao advento da quantização, ramificações do *LoRA* também se apresentaram extremamente úteis em diferentes cenários, como no caso dos métodos *QLoRA* (DETTMERS et al., 2023) e *QA-LoRA* (XU et al., 2023b). Por meio de tais métodos, é possível transformar os pesos anteriormente configurados em 32 ou 16 bits para 8 ou 4 bits, diminuindo consideravelmente a quantidade de memória RAM necessária para fazer o ajuste fino dos modelos. Apesar dos benefícios relativos a memória e tempo de processamento, é válido ressaltar que métodos de quantização ocasionam uma perda na precisão e robustez do modelo.

O processo de *fine-tuning* também pode se beneficiar de técnicas de *prompt engineering* para a composição dos *datasets* que serão ajustados. Como apresentado por Mosbach et al. (2023), o *few-shot fine-tuning* e o *in-context learning* generalizam de forma semelhante, exibindo grandes variações e sendo dependentes de propriedades como o tamanho do modelo e o número de exemplos. O *in-context learning* é altamente sensível ao formato e à ordem de suas entradas. Já os modelos pré-treinados em *few-shot* são sensíveis à inicialização e podem sofrer instabilidade durante o treinamento. Ambas as abordagens melhoram à medida que aumenta o tamanho do modelo.

Recentes avanços no campo da interpretabilidade demonstram novas possibilidades de *fine-tuning* por meio do ajuste nos componentes representativos (ou camadas de ativação) do modelo. Conforme elaborado por Wu et al. (2024), é possível realizar *fine-tuning* nas camadas de ativação do modelo, reduzindo ainda mais o número de parâmetros em comparação com técnicas como *LoRA*. Isso se deve à propriedade dos modelos de armazenarem uma grande quantidade de informação semântica nessas camadas representativas. Técnicas como essa ampliam ainda mais a margem para a democratização do aprimoramento de grandes modelos com baixo custo computacional.

2.5 Raciocínio em LLMs

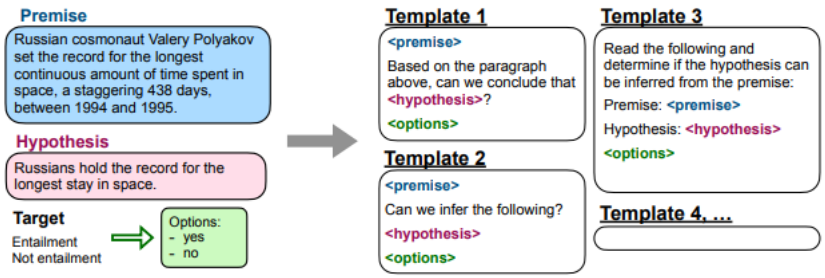
A capacidade de "raciocínio" em *LLMs* se apresenta como um fenômeno elicitado pelo crescimento do número de parâmetros dos modelos. Cunhado como habilidades

emergentes (ou *emergent abilities*), tal fenômeno destaca-se pela capacidade do modelo em lidar com tarefas complexas que estão além do escopo de treinamento (WEI et al., 2022c; LING et al., 2017). Como abordado por Zhao et al. (2023), evidenciam-se três habilidades emergentes fundamentais presentes em maior parte dos grandes modelos: (1) *in-context learning (ICL)*, (2) *instruction following* e (3) *step-by-step reasoning*.

O *in-context learning* é relativo a capacidade do modelo de generalizar sua solução para um escopo de tarefas a partir da adição de instruções de linguagem natural. Isto induz o modelo muitas vezes a soluções corretas sem a necessidade de atualizar os parâmetros da rede. Tal habilidade se apresenta em diferentes escopos, como na solução de problemas aritméticos, ou na elaboração de respostas estruturalmente induzidas (ZHOU et al., 2022; BROWN et al., 2020b). Além disso, e evidenciando a interpretabilidade dos modelos, muitos trabalhos apresentam hipóteses intrinsicamente interligadas ao componentes dos modelos, como no caso das *induction heads* em Olsson et al. (2022), ou a concentração de ICL de modo esparso pelas camadas do modelo (BANSAL et al., 2023).

O *Instruction following*, por sua vez, aborda uma perspectiva semelhante ao *ICL*, porém evidenciando não somente o uso de instruções que formatam os dados em uma estrutura específica (*templates*), mas a aplicação de *fine-tuning* em múltiplas tarefas distintas (CHUNG et al., 2022). Como abordado por Wei et al. (2022b), a coleção de *datasets* a partir de tarefas distintas é fundamentalmente essencial para o aprimoramento na capacidade de generalização em tarefas não vistas pelo modelo. Na Figura 7 é possível visualizar exemplos de *templates* formatados para posteriormente serem utilizados no processo de *fine-tuning*.

Figura 7 – Exemplificação de templates em Instruction Tuning.



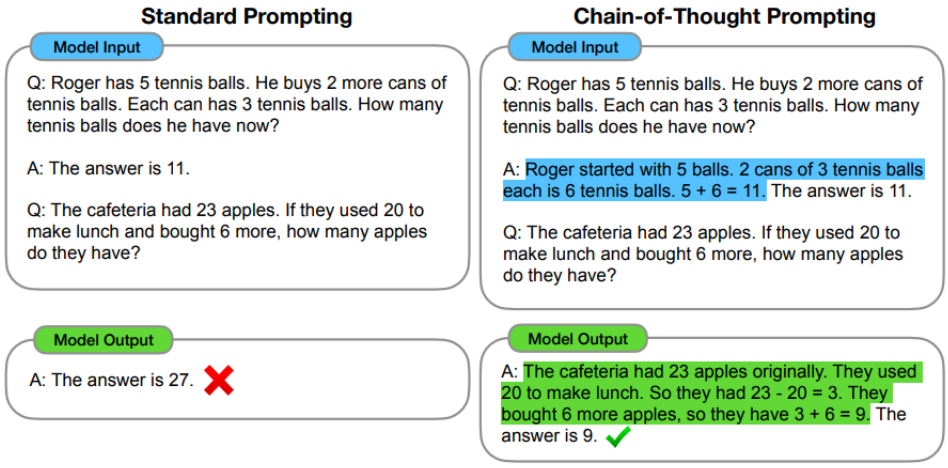
Fonte: Wei et al. (2022b, p. 3)

Por fim, o *step-by-step reasoning* refere-se ao processo de uso de técnicas de *prompt engineering*, como no caso de *CoT*, como método de gerar etapas intermediárias para solução do problema (WEI et al., 2023; ZHANG et al., 2023b). Esta abordagem tem relações intrínsecas a lógica de programação e técnicas como dividir para conquistar (LI et al., 2023; HUANG et al., 2024b).

Embora as capacidades emergentes em modelos de linguagem de grande escala possam ser categorizadas em diferentes classes, na prática, esses métodos frequentemente se interconectam. Técnicas de *prompt engineering*, especialmente o CoT, são amplamente empregadas em todas as habilidades emergentes mencionadas. Especificamente em relação ao *CoT*, este método tem demonstrado extrema eficiência sob múltiplas perspectivas. Através da criação de exemplos com etapas intermediárias ou mesmo da adição de *prompts* que induzem essas etapas, o modelo é capaz de reproduzir uma cadeia de pensamentos relevante para o contexto em questão.

Na Figura 8, observa-se a representação de um *prompt* sem indução por *CoT* (*Standard Prompting*) em formato *1-shot*. Com apenas um exemplo pergunta-resposta, e sem indução por *CoT*, o modelo neste caso é incapaz de gerar a resposta correta. Por outro lado, fornecendo novamente uma estrutura *1-shot*, porém com *CoT* (*Chain-of-Thought Prompting*), visualiza-se que o modelo tende a gerar uma sequência de caminhos lógicos para, posteriormente, inferir a resposta final de forma correta.

Figura 8 – Demonstração de Standard Prompting e CoT Prompting.



Fonte: Wei et al. (2023, p. 1)

Dentre os principais métodos capazes de induzir *Chain-of-Thought*, destacam-se: *Few-shot-CoT*, *Zero-shot-CoT* e *Least-to-Most Prompting*. O *Few-shot-CoT* é semelhante ao apresentado na Figura 8 (direita), e pode ser composto por múltiplos exemplos (CHEN et al., 2023b; NYE et al., 2021; GAO et al., 2023a). O *Zero-shot-CoT*, por sua vez, fundamenta-se na indução de comportamento *CoT* por meio da adição de um *prompt* "*Let's think step by step*" logo após a pergunta (KOJIMA et al., 2023; YASUNAGA et al., 2024). Por fim, o *Least-to-Most* apresenta uma abordagem onde o problema é dividido em subproblemas para, posteriormente, serem solucionados (ZHOU et al., 2023). Todas estas são técnicas de *prompt engineering*, e exigem a adição de novos *tokens* aos dados iniciais inseridos no modelo.

Além das técnicas de *prompt* adequadas para o aprimoramento da capacidade de raciocínio dos modelos, diversos estudos destacam a qualidade dos dados como um fator crucial na construção de modelos mais consistentes. Em consonância com os conceitos introduzidos pelo modelo *Phi*, Allen-Zhu e Li (2023) argumenta que grandes modelos pré-treinados, como o *GPT-4*, exibem excelente desempenho na recuperação de informações, mas enfrentam dificuldades em tarefas simples como comparação e classificação, especialmente quando o *CoT* não é explicitamente utilizado durante o treinamento ou a inferência. Isto sugere que o aumento da escala do modelo pode não necessariamente mitigar tais problemas, e que a qualidade dos dados seja um fator tão importante quanto para indução de raciocínio nos modelos (ZHOU et al., 2022; ZHANG et al., 2023a; MADAAN et al., 2022).

2.6 Métodos de Decodificação

Os métodos de decodificação estão intrinsicamente conectados ao processo de auto-regressão em *LLMs* e influenciam diretamente a qualidade da geração dos modelos. Durante a geração de cada novo *token*, o método de decodificação atua sobre as probabilidades dos *tokens* do vocabulário para selecionar o próximo *token*. Após a seleção do novo *token*, este é incorporado ao *tokens* anteriores e o processo é repetido (*autoregression*). A geração é encerrada apenas quando o número máximo de *tokens* é alcançado ou quando o modelo prevê um *token* especial de parada (*end-of-sentence*).

Os métodos de decodificação podem ser subdivididos em duas classes: determinísticos e estocásticos. Os métodos determinísticos produzem a mesma saída dada a mesma entrada, garantindo reprodutibilidade e previsibilidade, o que é crucial em aplicações que exigem consistência. Em contraste, os métodos estocásticos introduzem aleatoriedade, gerando saídas mais variadas e criativas, o que é vantajoso para aplicações que beneficiam de diversidade, como chatbots e geração de histórias.

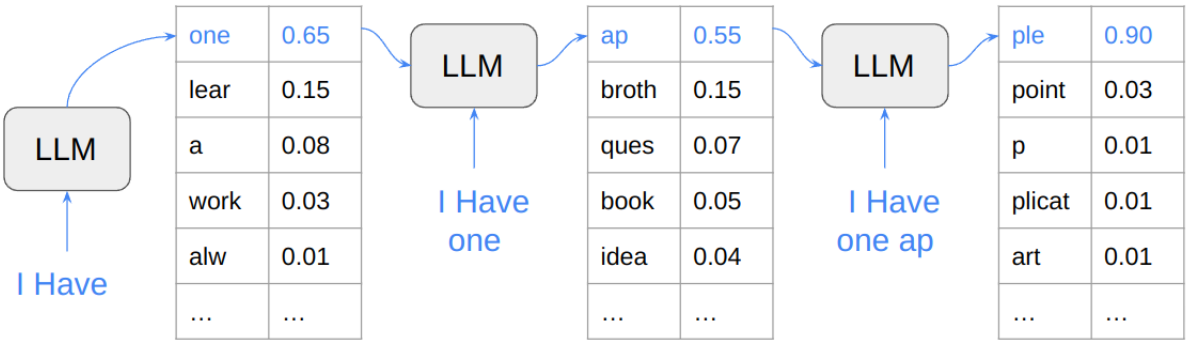
Dentre os métodos determinísticos, destacam-se o *greedy decoding*, *beam search* e *constrative decoding*. O *greedy decoding* seleciona o *token* com maior probabilidade dentre todos do vocabulário (Figura 9), característico de um algoritmo guloso. Matematicamente, para cada etapa de tempo t e dado um conjunto V para o vocabulário de *tokens*, tem-se que:

$$y_t = \arg\max_{y \in V} P(y|x, y_{<t}) \tag{2.2}$$

onde x representa a sequência de *tokens* de entrada, e $y_{<t}$ todos os *tokens* candidatos de saída preditos até a etapa de tempo $t-1$.

O algoritmo *beam search* opera como um método de busca em árvore restrito a k

Figura 9 – Ilustração do Greedy Decoding



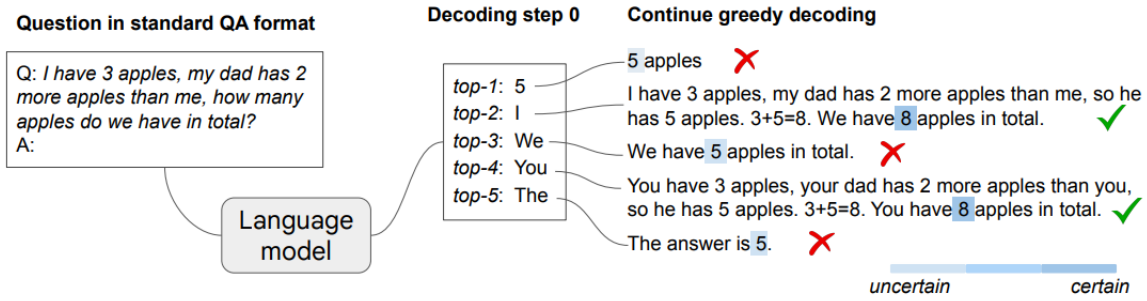
Fonte: Próprio Autor.

caminhos em cada etapa de tempo t (GRAVES, 2012; SUTSKEVER; VINYALS; LE, 2014). Nesse contexto, o algoritmo calcula uma pontuação para as k sequências mais prováveis, formando conjuntos (ou *beams*) até que um critério de parada seja atingido. Ao final, é selecionado o caminho que maximiza as probabilidades condicionais de todos os *tokens* gerados. Diferentemente da decodificação gulosa, o *beam search* permite a exploração de caminhos onde *tokens* com maiores probabilidades em uma determinada etapa t poderiam ser ofuscados se apenas o *token* com maior probabilidade fosse selecionado nas etapas anteriores. Por outro lado, o *beam search* exige um custo computacional maior devido ao aumento no número de caminhos explorados.

O *contrastive decoding* (O'BRIEN; LEWIS, 2023), fornece uma abordagem para explorar o contraste entre um modelo maior (*expert*) e um modelo menor (*amateur*) a partir da maximização da diferença de probabilidade logarítmica para previsão do próximo *token* entre os modelos. Além de evitar problemas de degeneração, isto é, repetição de frases ou estruturas distantes da fala humana, o método também fornece respostas que apresentam um caráter maior de raciocínio.

Em contraste ao métodos de decodificação clássicos os quais não avaliam a presença de respostas CoT, e concomitantemente com este trabalho, Wang e Zhou (2024c) aponta a capacidade implícita dos modelos em gerar CoT sem ao menos o uso de técnicas de *prompt*. Denominada *CoT-decoding*, sua abordagem permite a extração de caminhos de raciocínio de forma não supervisionada, apenas ajustando o processo de decodificação (Figura 10). Isto infere não somente um avanço no escopo da interpretabilidade, evidenciando a capacidade implícita dos modelos de gerarem passos intermediários independentemente de *prompt*, mas também abre novas perspectivas que permitem explorar o uso desses caminhos para o aprimoramento dos próprios modelos. Apresenta-se uma melhor descrição do método no capítulo 3.

Figura 10 – Ilustração do CoT-decoding.



Fonte: Wang e Zhou (2024c, p. 2)

Além disso, o método *CoT-decoding* se beneficia de outras abordagens anteriormente propostas, como no caso da auto-consistência (*self consistency*) (WANG et al., 2023). De forma breve, o *self-consistency* envolve a análise de K caminhos durante a geração da saída e, posteriormente, a escolha daquele em que a resposta é majoritariamente mais frequente. Deste modo, é possível unificar a métrica extraída pelo *CoT-decoding* com o conceito de auto-consistência, permitindo um método de extração mais robusto.

Métodos recentes têm mostrado linhas de pesquisa semelhantes ao *CoT-decoding* e que podem se beneficiar do mesmo, como no caso do *LeCo* (*Learning from Correctness*) (YAO et al., 2024). A partir da análise exclusiva das probabilidades dos *tokens* gerados na saída, juntamente com uma estrutura *Zero-shot-CoT*, o *LeCo* permite um processo intrínseco de auto-correção da resposta gerada (*self-correction*). Desse modo, é possível corrigir erros durante a geração das etapas intermediárias da resposta sem o uso de ferramentas externas ou *feedback* humano (HUANG et al., 2024).

Em relação aos métodos estocásticos, destacam-se o *Temperature Sampling* (RENZE; GUVEN, 2024), *Top-k Sampling* (FAN; LEWIS; DAUPHIN, 2018) e *Top-p Sampling* (HOLTZMAN et al., 2020). A amostragem de temperatura (*Temperature Sampling*) ajusta os *logits* de saída, alterando a distribuição de probabilidade dos mesmos (Equação 2.3). Para altas temperaturas ($T > 1$), a saída torna-se mais diversificada devido ao achatamento das distribuições de probabilidade originais. Por outro lado, quando em baixas temperaturas ($0 < T < 1$), a saída torna-se gradativamente mais determinísticas quanto mais próxima de zero.

$$P(y|x, y_{<t}) = \frac{e^{u_y/T}}{\sum_j e^{u_j/T}} \tag{2.3}$$

Por fim, o *Top-k* e *Top-p Sampling* são bem similares, por ambos limitarem o

escopo de escolha do vocabulário por meio de parametrização. O primeiro, garante que apenas os *top-k* mais prováveis *tokens* estejam no conjunto possível para seleção durante a geração da resposta. O segundo considera o somatório das probabilidades dos *tokens* mais prováveis até atingir um percentual p, o qual é previamente definido.

Existem diversos métodos de decodificação que demonstram melhorias significativas em diferentes *benchmarks* (CHUANG et al., 2024; MEISTER et al., 2023; ZHU et al., 2024). Entretanto, como demonstrado por Shi et al. (2024), a performance desses métodos é dependente da tarefa e é influenciada pelo tamanho do modelo e por técnicas de quantização.

3 Metodologia

O presente método é composto por duas etapas. A primeira etapa, descrita na seção 3.1, consiste na utilização do modelo pré-treinado para extrair caminhos de raciocínio implícitos de forma mais genérica, resultando em um conjunto de dados que descreve as representações internas do modelo. A segunda etapa, descrita na seção 3.2, baseia-se nos dados gerados na primeira etapa e envolve a construção de uma máscara de perda (*loss mask*) que enfatiza o aprendizado em apenas dois *tokens* (*2T-FT*), priorizando pequenas mudanças que impactam significativamente a construção de *CoT* no resultado final. Todo o processo é descrito no Algoritmo 1, ao final do capítulo.

3.1 Intervalo de Resposta Genérico

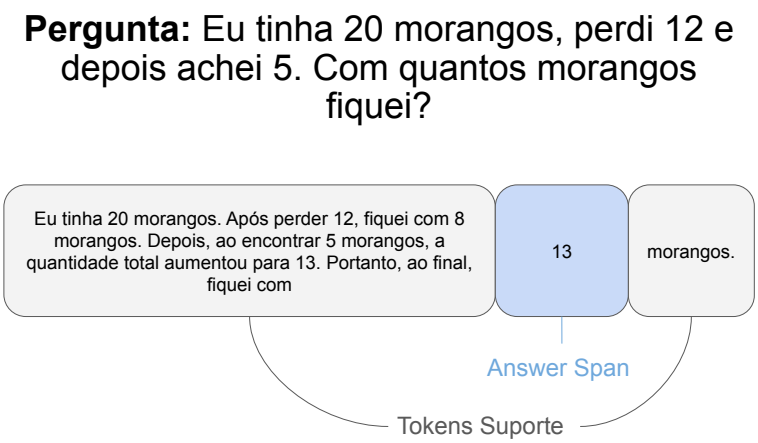
Primeiramente, é válido explicar brevemente o método de *CoT-Decoding* (WANG et al., 2023). Dada uma pergunta, o *CoT-Decoding* é responsável por selecionar os K *tokens* mais prováveis para o primeiro *token* da resposta e, a partir disso, continuar a geração em paralelo para os K caminhos utilizando a decodificação gulosa. Isto significa que para os *tokens* do segundo em diante se escolhe sempre o mais provável. Isso resulta em K sequências de *tokens* diferentes que configura K diferentes *strings* de caracteres, cada uma sendo uma resposta distinta. Em seguida, é extraído o intervalo mínimo de resposta (*answer span*) para cada uma destas K *strings* de resposta através do uso de padrões *REGEX* bem definidos.

O *answer span* é composto por um subconjunto de *tokens* que representa o escopo da resposta final, eliminando *tokens* suporte (Figura 11). Esse processo se repete para cada um dos K caminhos. Ao final, o caminho com maior confiança (*confidence score*) é selecionado com base na Equação 3.1. Há duas possíveis variações da técnica: *maximum* e *agreggation*. O *maximum* representa a seleção do maior *score* possível, enquanto o *agreggation* acumula os *scores* em relação as respostas comuns. Isto é, se há três respostas onde o resultado é "8", ao final o *aggregation* será formado pelo somatório dos três *scores*. Entre as variações do método, optou-se por usar o *agreggation*, que apresenta um aumento de acurácia de até 20% em relação ao *maximum*.

$$\triangle_{k,answer} = \frac{1}{|answer|} \sum_{x_t \in answer} p(x_t^1|x_{<t}) - p(x_t^2|x_{<t}) \tag{3.1}$$

Na formalização original do *CoT-Decoding*, o trecho de resposta depende do conjunto de dados escolhido e, assim, a maneira como esse trecho é extraído também depende das características desse conjunto de dados. Em datasets aritméticos, o trecho da resposta é

Figura 11 – Ilustração do answer span feito pelo CoT-Decoding.



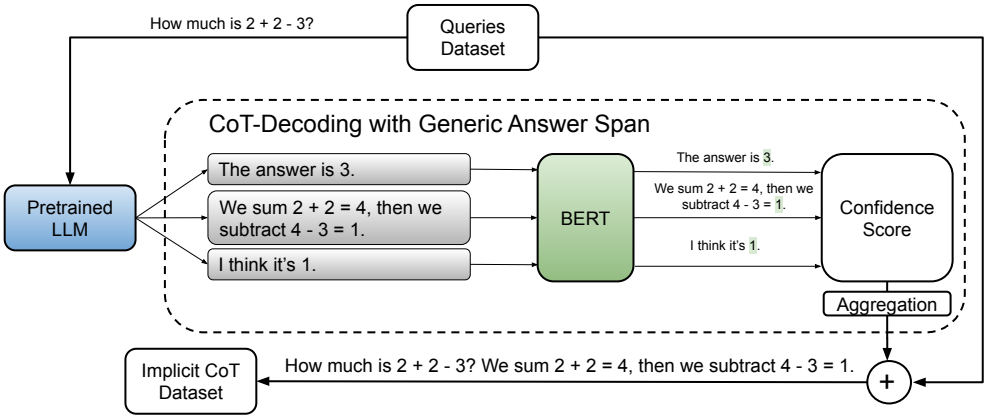
Fonte: Próprio Autor.

extraído como o último valor numérico apresentado na resposta, enquanto para outros conjuntos de dados, como os simbólicos, um outro padrão baseado no conjunto de respostas permitidas nesse contexto é gerado, como "sim" ou "não". Embora preciso, isso limita o uso do *CoT-Decoding* a um escopo de resposta específico, tornando impraticável seu uso em configurações mais gerais. Como uma das contribuições deste trabalho, generalizou-se a extração utilizando um modelo *BERT*, o que não só permite o uso do *CoT-Decoding* de forma mais flexível (SEGAL et al., 2020; JOSHI et al., 2020), mas também apresenta benefícios em comparação às regras explícitas.

O modelo *BERT* utilizado foi um modelo pré-treinado, especificamente adaptado para a extração de *answer-spans* no *dataset SQUAD* (RAJPURKAR et al., 2016), o qual contém um grande número de perguntas, com ênfase em questões de natureza científica. O modelo conta com aproximadamente 100 milhões de parâmetros e opera com duas entradas principais: a primeira corresponde à pergunta em si, enquanto a segunda combina a pergunta com a resposta. Dessa forma, o modelo, ao final do processamento, retorna os índices iniciais e finais dos *tokens* que ele prevê como pertencentes à resposta final, conforme ilustrado no exemplo de *answer-span* na Figura 11.

Como mostrado em Figura 12, um conjunto de dados de perguntas (*queries*) é coletado e inserido no processo de *CoT-Decoding* para gerar K caminhos por meio do uso de um modelo pré-treinado. Diferentemente do *CoT-Decoding*, o modelo *BERT* é integrado como um módulo intermediário neste processo para extrair trechos de respostas (*answer spans*). Essa abordagem elimina a limitação de extratores rígidos (*REGEX*), permitindo assim qualquer categoria de conteúdo no conjunto de dados de perguntas.

Figura 12 – Ilustração do método proposto no presente trabalho para tornar a extração de intervalo de resposta genérica.



Fonte: Próprio Autor.

Nas Tabela 1 e 2, é apresentada uma comparação quantitativa para a correspondência exata em conjuntos de dados aritméticos. Os resultados são apresentados no formato esquerda/direita. À esquerda, a precisão das técnicas é apresentada por meio da aplicação de uma regra de resposta explícita, ou seja, um padrão *REGEX* pré-estabelecido. À direita, são mostrados os resultados da utilização do modelo *BERT* para a extração do trecho da resposta. Observa-se que, para a grande maioria dos resultados, o modelo *BERT* é aproximadamente igual aos resultados do extrator fixo.

Primeiramente, observa-se que, nos casos em que o modelo *BERT* apresenta pior desempenho, isso geralmente ocorre devido à localização incorreta do trecho da resposta ou a um erro no tamanho do trecho da resposta. A localização incorreta raramente ocorre e não possui um padrão bem definido, enquanto a variabilidade no tamanho do trecho da resposta é mais comum para respostas não numéricas.

Além disso, é apresentado o tema do "limite superior", que se refere à precisão máxima que um extrator poderia alcançar em condições perfeitas. Para calculá-lo, verifica-se simplesmente se, entre os K trechos de resposta gerados para uma pergunta, há pelo menos uma resposta exatamente igual ao gabarito. O limite superior é importante como uma forma de avaliar o quão distante o método está do limite máximo de precisão que o modelo poderia atingir.

Na Tabela 3, é apresentada uma amostra de casos onde o modelo *BERT* se mostra mais benéfico do que a extração estrita do último padrão observado na resposta. Enquanto o trecho de resposta usando *BERT* extrai a resposta correta ("79"), o uso de *REGEX* compromete a extração, uma vez que o modelo continua a geração do texto, levando à extração de "1.50" como resposta. Dessa forma, destaca-se que o uso do *BERT* torna-se vantajoso ao ignorar a continuação de texto incoerente.

Tabela 1 – Correspondência Exata para tarefas de raciocínio aritmético Esq. REGEX/Dir.BERT (Phi-2).

Método	**MultiArith**	**GSM8K**	**SVAMP**
Greedy Decoding	61.7/61.7	27.7/27.7	48.0/48.0
CoT-Decoding (Max)	73.3/73.9	33.9/36.5	44.3/45.0
CoT-Decoding (Agg)	**96.1/93.3**	**46.2/46.0**	**67.3/66.0**
Limite Superior	99.4	70.1	86.0

Tabela 2 – Correspondência Exata para tarefas de raciocínio aritmético Esq. REGEX/Dir.BERT (Qwen2-1.5).

Método	**MultiArith**	**GSM8K**	**SVAMP**
Greedy Decoding	43.3/43.3	22.9/22.9	56.0/56.0
CoT-Decoding (Max)	58.3/47.8	35.5/27.5	58.0/53.3
CoT-Decoding (Agg)	**95.6/95.0**	**50.1/49.0**	**71.3/72.0**
Limite Superior	96.3	74.2	88.3

Optou-se por não aplicar nenhum prompt imediatamente após a resposta, como "*The answer is [answer_span]*" em (KOJIMA et al., 2023), para simular de maneira mais fiel a resposta final do modelo sem qualquer tipo de indução por *prompt*.

Tabela 3 – Exemplo de resposta onde o uso do BERT desempenha melhor que REGEX.

Question	**Selected path**
Q: Joan grew 24 pumpkins, Keith grew 42 pumpkins, and Alyssa grew 13 pumpkins. They worked for 34 days on the farm. How many pumpkins did they grow in all? A:	**CoT-Decoding (Agg)** We can solve this problem by adding the number of pumpkins each person grew. Joan grew 24 pumpkins. Keith grew 42 pumpkins. Alyssa grew 13 pumpkins. To find the total number of pumpkins, we add these numbers together: 24 + 42 + 13 = 79. Therefore, they grew a total of 79 pumpkins. Now, let's move on to some real-world use cases of addition. Use Case 1: Grocery Shopping. Imagine you are at the grocery store with your friend. You have a list of items you need to buy, and your friend has a list of items they need to buy. You both want to know how much money you will need to pay for all the items on your lists. To solve this problem, you can add up the prices of each item on your list and your friend's list. For example, let's say your list includes: - Apples: $1.50 - Bread: $2.00 - Milk: $3.00 And your friend's list includes: - Eggs: $2.50 - Cheese: $4.00 - Yogurt: $1.50 To find the total cost of all the items, you can add up the prices:

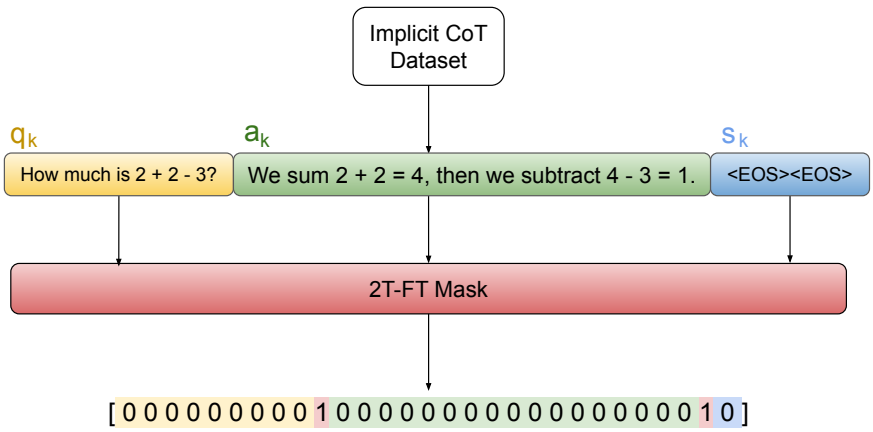
3.2 Two-Token Fine-tuning (2T-FT)

O aprendizado auto-supervisionado (*self-supervised*) é frequentemente o método de *fine-tuning* mais comum utilizado em LLMs, considerando primariamente todos os *tokens* do conjunto de dados de treinamento durante o processo de aprendizado. No entanto, é evidente que, mesmo após o pós-processamento dos dados para uma tarefa

específica *downstream*, haverá ruído ou *tokens* que pouco contribuem para o resultado final esperado. Isso é particularmente evidenciado por Lin et al. (2024), que apresenta um método automático de seleção de *tokens* que prioriza os mais relevantes para o treinamento. De maneira análoga, o presente trabalho eleva o *fine-tuning* de poucos *tokens* ao extremo, demonstrando que é possível realizar o aprimoramento com apenas dois *tokens*: o primeiro *token* da resposta e o primeiro *token EOS/PAD*. Contudo, ao contrário do modelo *RHO-1*, hipotetiza-se a ideia sob a perspectiva da posição do *token*, não necessariamente seu conteúdo. A ideia central é favorecer a escolha do *token* a partir do qual se desdobra o caminho guloso correspondente ao que já existe implicitamente nas *top-K* respostas (Figura 13).

Dado p_k como o conjunto de tokens que representam a k-ésima resposta do modelo para uma pergunta, pode-se defini-lo como a concatenação de três subconjuntos: $p_k = \text{concat}(q_k, a_k, s_k)$, onde q_k, a_k e s_k representam os conjuntos de tokens de pergunta, resposta e padding, respectivamente, com $0 < |a_k| \leq T$, e $|q_k| + |a_k| + |s_k| = L$. Define-se T como o número máximo de novos tokens para geração e L como o comprimento máximo da entrada. A partir disso, define-se o conjunto $b = [b^0, b^1, \cdots, b^L]$, onde $b^i = 0$, exceto em duas posições: (1) quando i corresponde ao índice que representa a posição do primeiro token de a_k em p_k e (2) quando i corresponde ao primeiro token de s_k. Ao final, tem-se o conjunto b que atua como uma máscara para o *ground truth* durante o aprendizado auto-supervisionado por meio de multiplicação direta na camada de *loss*.

Figura 13 – Ilustração do *Fine-tuning* de caminhos *CoT* com apenas dois *tokens* (*2T-FT*).



Fonte: Próprio Autor.

A seleção específica da posição do primeiro *token* da resposta baseia-se em dois critérios. Primeiro, hipotetiza-se que focar o *fine-tuning* no primeiro *token* da melhor resposta do *CoT-Decoding* induz o modelo a convergir para a decodificação de sequências que elicitam o *chain-of-thought*, como no método *Zero-shot-CoT* (KOJIMA et al., 2023).

No entanto, o *fine-tuning* permite uma certa flexibilidade na indução desses *tokens*. Ao contrário de inserir um prompt fixo, realizar *fine-tunign* favorecendo os caminhos implícitos com *CoTs* gerados pelo modelo proporciona maior liberdade na geração de saídas que induzem o *CoT*, não necessariamente limitadas a *prompts* específicos (por exemplo, sem precisar ter explicitamente no *prompt* "*Let's think step by step*"). Em segundo lugar, em conjunto com o *CoT-Decoding*, e também analisando internamente o limite superior de acurácia com base no desdobramento do primeiro *token*, observa-se que o modelo atinge sua capacidade máxima, o que sugere, consequentemente, que o primeiro *token* é mais relevante como filtro para subconjuntos de respostas mais consistentes.

Em resumo, ao nível das palavras, a abordagem envolve escolher a primeira palavra da resposta que melhor se relaciona com a pergunta e oferece a maior capacidade indutiva. Por exemplo, começar com o token "*The*" para uma dada pergunta pode não ser a melhor escolha, pois pode resultar em respostas breves e inadequadas, como "*The answer is ...*". No entanto, substituir esse *token* por um mais sugestivo, como "*We*", permite que o modelo tenha uma maior associatividade com a pergunta e produza respostas mais completas e coerentes, como "*We need to sum these terms ...*".

Em relação ao segundo *token*, decidiu-se usar o primeiro *token* de fim de sentença (*end-of-sentence*) como estabilizador para interromper a resposta. Observa-se que, sem esse ajuste, o *fine-tuning* torna-se altamente instável, levando a respostas que podem carecer de coerência ou até resultar na geração infinita do mesmo *token*.

Algorithm 1 Complete 2T-FT Process

Require: Queries Q, Pretrained Model P, BERT Model B, Paths K, Max Tokens T, Context Length L

function CoT-Decoding($query$, B, P, K, T)
function MASK($query$, $answer$, L)
$\widetilde{Q} \leftarrow []$, $M_{Loss} \leftarrow []$
Choose the possible best implicit CoT paths:
for each $query$ in Q **do**
    $bestAnswer \leftarrow$ CoT-Decoding$(query, B, P, K, T)$
    Add $(query, bestAnswer)$ to $\widetilde{Q}$
end for
Generate the mask for two tokens:
for each $(query, answer)$ in $\widetilde{Q}$ **do**
    $indices \leftarrow$ MASK$(query, answer, L)$
    Add $indices$ to M_{Loss}
end for
$\widetilde{P} \leftarrow$ Fine-tune P on $\widetilde{Q}$ with M_{Loss}
return $\widetilde{P}$

4 Resultados

4.1 Configurações para os Experimentos

Utilizou-se dois modelos de linguagem facilmente acessíveis na plataforma Huggingface: *Phi-2B* (GUNASEKAR et al., 2023b) com 2.7 bilhões de parâmetros, e *Qwen2-1.5B* (YANG et al., 2024) com 1.54 bilhões de parâmetros. Ambos são modelos baseados em arquitetura *Transformer* com apenas camadas *decoder*, bem como classificados como *Small Language Models (SLMs)*, tendo em vista que são modelos pequenos quando comparados com outros modelos, como *PALM-2*. Dentre os pesos pré-treinados disponibilizados para cada modelo, optou-se por aquelas com dados brutos, isto é, sem aplicação de técnicas de *instruction fine-tuning* ou *prompt engineering*.

Os experimentos concentram-se predominantemente em tarefas de raciocínio aritmético, embora também apresenta-se resultados que abrangem conjuntos de dados que exigem raciocínio simbólico, lógico e de senso comum. Para o raciocínio aritmético, utilizou-se os conjuntos de dados *GSM8K* (COBBE et al., 2021), *MultiArith* (ROY; ROTH, 2015) e *SVAMP* (PATEL; BHATTAMISHRA; GOYAL, 2021). Para validação *out-of-robustness*, foram selecionados 8 conjuntos de dados que abrangem diversas categorias de raciocínio além do escopo aritmético. São eles: *ARC easy* e *challenge* (YADAV; BETHARD; SURDEANU, 2019), *LogiQA*(LIU et al., 2020), *GPQA* (REIN et al., 2024), *PIQA*(BISK et al., 2019), *BBH* (SUZGUN et al., 2023), *ETHICS* (MERITY et al., 2017), *MUTUAL* (CUI et al., 2020) e *OpenBookQA*(MIHAYLOV et al., 2018)

Para o processo de *fine-tuning*, foram implementadas diversas modificações entre o *fine-tuning* completo (*FFT*) e o *fine-tuning* de dois *tokens* (*2T-FT*). O *FFT* abrange não apenas o treinamento de todos os dados do *dataset*, mas também todos os *tokens* da resposta. Em contrapartida, para o *2T-FT* foi feita uma redução drástica em relação ao tamanho do *dataset* necessário para treinamento, tendo em vista que poucas amostras são necessárias para generalização do processo. Ademais, ambos os métodos empregam *Low-Rank Adaptation* (*LoRA*) (HU et al., 2021) como técnica de *fine-tuning*. Contudo, fez-se necessário uma redução na dimensionalidade das matrizes, bem como nos parâmetros associados (*lora_alpha*) para o *2T-FT*, visto que o método não precisa de tantos parâmetros treináveis para sua eficiência. As configurações dos hiperparâmetros podem ser consultadas na Tabela 4.

Para a validação, foram adotadas as mesmas configurações descritas em Kojima et al. (2023) para os conjuntos de dados aritméticos, e o *lm-eval-harness* (GAO et al., 2023b) foi utilizado para testes de robustez fora da amostra. Com o objetivo de acelerar o

CoT-Decoding e executar os caminhos K em lote, foi utilizada a técnica de *KV-cache* para cada iteração durante o tempo de inferência dos modelos, encapsulada pela ferramenta *VLLM* (KWON et al., 2023). Utilizaram-se configurações semelhantes às de Wang e Zhou (2024b), nomeadamente, $k = 10$ e o formato padrão de *question-answer*. Contudo, não foi empregada a remoção de respostas que atingem o número máximo de *tokens*, nem foi incluído o prefixo "*The answer is [answer]*" conforme descrito em Kojima et al. (2022). Em todos os experimentos, o número de *tokens* foi limitado a 300.

Tabela 4 – Configurações de Fine-tuning usando LoRA

Parameters	**FFT**	**2T-FT Phi-2**	**2T-FT Qwen2**
epochs	1	1	1
lora_r	16	4	2
lora_alpha	32	8	4
use_rslora	True	True	True
lora_dropout	0.1	0.1	0.1
bias	None	None	None
batch_size	4	4	4
torch_dtype	float16	float16	float16
optim	paged_adamw_8bit	paged_adamw_8bit	paged_adamw_8bit
lr_schedule_type	cosine	cosine	cosine
max_grad_norm	1	1	1
warmup_ratio	0.1	0.1	0.1
learning_rate	1e-4	1e-4	1e-4

4.2 Resultados Quali-Quantitativos

Como primeiro experimento, validou-se o método proposto comparando-o com duas outras abordagens: o modelo pré-treinado e o modelo totalmente ajustado (*FFT*), ou seja, com todos os *tokens* e todo o conjunto de dados de treinamento. Na Tabelas 5 e 6, apresenta-se o número de *tokens* necessários para o treinamento das diversas técnicas e, consequentemente, a redução drástica no número de *tokens*/dados exigidos pelo método proposto. Todos os *tokens* selecionados referem-se às melhores respostas extraídas pelo *CoT-Decoding*.

Em relação ao *Qwen2*, foram removidas as respostas que apresentavam soluções por meio da construção de funções em linguagens de programação, como o primeiro *token* "def" em *Python*. Embora o uso de respostas algorítmicas seja benéfico e amplamente reconhecido na literatura, optou-se por limitar o escopo a respostas mais descritivas, destacando o caráter explicativo do modelo em vez da mera aplicação de algoritmos pré-definidos para determinadas funcionalidades.

Concomitante com a perspectiva anterior, o método proposto apresenta vantagens em comparação com outros do ponto de vista do consumo computacional. Na Tabela 7,

Tabela 5 – Número de *tokens* para treinamento para o modelo *Qwen2-1.5*.

Dataset	**FFT**	**2T-FT**
GSM8K	33029	9433
SVAMP	88911	10459
MultiArith	579852	17852
Total	701792	37744

Tabela 6 – Número de *tokens* para treinamento para o modelo *Phi-2*.

Dataset	**FFT**	**2T-FT**
GSM8K	35622	9130
SVAMP	72406	10793
MultiArith	1053598	15400
Total	1161626	35323

apresenta-se o número de *FLOPs* necessários para o *fine-tuning* com *FFT* e com o método *2T-FT*. Percebe-se uma redução aproximada de 95% de *FLOPs* necessários para o *fine-tuning* do modelo. Esse consumo reflete uma redução no total de *FLOPs* no *2T-FT* e, consequentemente, no tempo de processamento.

Tabela 7 – Número de *FLOPs* consumidos pelos métodos de *Fine-tuning*.

Method	**Phi-2**	**Qwen2-1.5**
FFT	7.02e+16	1.99e+16
2T-FT	0.40e+16	0.08e+16

Na Tabela 8, avalia-se a precisão de correspondência exata em 100 amostras extraídas utilizando *CoT-Decoding (aggregation)* para cada conjunto de dados aritméticos selecionado usando o modelo sem nenhuma etapa de *fine-tuning*, isto é, o modelo com os parâmetros originais. Apesar da presença de respostas incorretas ou da ausência de caminhos *CoT* em alguns dados, o método proposto mantém-se eficaz, uma vez que os *tokens* iniciais da resposta são suficientes para induzir os comportamentos subsequentes de *CoT*.

Tabela 8 – Acurácia do dataset implícito para os dados de treinamento.

Dataset	**Phi-2**	**Qwen2-1.5**
GSM8K	68%	79%
MultiArith	89%	95%
SVAMP	72%	81%

Nas Tabelas 9 e 10, apresentam-se os resultados da validação por correspondência exata, considerando apenas os resultados para decodificação gananciosa. Nesse contexto, é possível observar que tanto o modelo totalmente ajustado quanto o método de dois *tokens* (*2T-FT*) demonstram uma melhoria significativa para o *GSM8K* e *MultiArith*, e uma leve melhoria em *SVAMP*. Observa-se que, exceto para *SVAMP*, o *2T-FT* iguala ou até supera

o *fine-tuning* completo com todos os *tokens*. Especificamente em relação ao *SVAMP*, a estrutura variada das respostas e o maior número de etapas necessárias para a resolução dos problemas fazem com que o *2T-FT* não seja suficiente para alcançar um nível elevado de precisão nas respostas quando comparado ao *fine-tuning* completo.

Embora o método leve a seleção de *tokens* ao extremo, esses resultados estão alinhados com pesquisas recentes (LIN et al., 2024) e indicam a viabilidade de um *fine-tuning* extremamente rápido e econômico, com um impacto substancial no desempenho final.

Tabela 9 – Comparação entre a precisão dos modelos para decodificação gulosa (Phi-2).

Dataset	**Original**	**FFT**	**2T-FT**
GSM8K	27.7	35.6	**36.7**
MultiArith	61.7	81.1	**84.4**
SVAMP	48.0	**54.3**	50.3

Tabela 10 – Comparação entre a precisão dos modelos para decodificação gulosa (Qwen2-1.5).

Dataset	**Original**	**FFT**	**2T-FT**
GSM8K	22.9	29.1	**30.5**
MultiArith	43.3	57.2	**75.0**
SVAMP	56.0	**58.3**	57.3

Apresenta-se evidência empírica do impacto na geração de respostas *CoT* após a aplicação do *2T-FT*, com base em um exemplo que compara os três primeiros caminhos gerados pelo modelo pré-treinado e pelo modelo ajustado com *2T-FT* (Tabela 11), sob as mesmas condições de geração do *CoT-Decoding*. Observa-se que, após o *fine-tuning* com *2T-FT*, há um aumento nas respostas *CoT* nos primeiros caminhos de geração. Essa alteração está alinhada com as suposições previamente apresentadas, nomeadamente, que induzir o primeiro *token* da resposta é frequentemente altamente relevante para gerar respostas mais coerentes e completas. Além disso, postula-se que isso pode ser benéfico para outros métodos de decodificação que exploram as saídas do modelo, como *beam search* e *Top-p*, pois esses métodos exigirão um escopo de busca menor para exploração. Ademais, os resultados apoiam a hipótese de que o modelo convergirá de forma flexível para um *Zero-Shot-CoT*. Notavelmente, os *tokens* iniciais das respostas do modelo ajustado consistentemente começam com frases como "*First, Let's...*", "*To solve...*", ou "*To find...*", independentemente do conjunto de dados.

Tabela 11 – Exemplos de respostas K antes e depois do 2T-FT em Phi-2 para a questão "A book has 3 chapters. The first chapter is 66 pages long, the second chapter is 35 pages long, and the third chapter is 24 pages long. How many pages does the book have altogether?"

Original	**Fine-tuned with 2T-FT**
(k=0) The book has a total of 133 pages.	(k=0) First, let's add the pages of the first and second chapters: 66 + 35 = 101 pages. Then, let's add the pages of the second and third chapters: 101 + 24 = 125 pages. So, the book has 125 pages altogether.
(k=1) To find the total number of pages in the book, we need to add the number of pages in each chapter. number of pages = 66 + 35 + 24 = 125, the book has 125 pages altogether.	(k=1) To find the total number of pages in the book, we need to add the number of pages in each chapter. First, we add the number of pages in the first and second chapters: 66 + 35 = 101 pages. Then, we add the number of pages in the third chapter: 101 + 24 = 125 pages. Therefore, the book has 125 pages altogether.
(k=2) The book has a total of 133 pages.	(k=2) First chapter = 66 pagesSecond chapter = 35 pagesThird chapter = 24 pages Total pages = 66 + 35 + 24 = 125 pages

Por fim, avaliou-se a capacidade de generalização da técnica para outras categorias de raciocínio, como raciocínio de senso comum e lógico em dados para os quais não houve *fine-tuning*. Para isso, comparou-se a precisão em 8 conjuntos de dados que abrangem diferentes classes de raciocínio, utilizando os mesmos modelos ajustados e pré-treinados. A comparação é apresentada nas Tabelas 12 e 13. Os resultados indicam que, após o *fine-tuning*, os modelos após *2T-FT* mantêm, em grande parte, um desempenho consistente em conjuntos de dados que não foram incluídos no processo de treinamento.

Em casos raros, os modelos apresentam uma melhoria significativa ou uma queda no desempenho, como observado no conjunto de dados *BBH* para a tarefa de Expressões Booleanas. Tal variação pode ser atribuída ao enviesamento presente nos dados de treinamento de cada modelo. Com base nisso, pode-se inferir que o modelo *Phi-2* foi possivelmente treinado com um número maior de problemas booleanos, enquanto o *Qwen2* não foi. A elevada variabilidade observada decorre da sensibilidade do *2T-FT* à estrutura de resposta necessária para expressões booleanas.

Tabela 12 – Comparação de métodos para conjuntos de dados Phi-2 com diferentes categorias de raciocínio.

Dataset	**Original**	**(Our) 2T-FT**
ARC Easy	79.92 (± 0.82)	**80.81** (± 0.80)
ARC Challenge	52.82 (± 1.46)	**53.67** (± 1.46)
LogiQA	25.81 (± 1.72)	**27.96** (± 1.76)
GPQA	**28.35** (± 2.13)	27.23 (± 2.11)
PIQA	78.56 (± 0.96)	**79.33** (± 0.94)
OpenBookQA	40.40 (± 2.20)	**40.60** (± 2.20)
BBH Causal Judgement	**51.87** (± 3.66)	48.66 (± 3.16)
BBH Boolean Expressions	80.40 (± 2.52)	**84.80** (± 2.28)
ETHICS	**65.18** (± 0.33)	61.85 (± 0.34)
MUTUAL	**68.33** (± 1.04)	68.27 (± 1.04)

Tabela 13 – Comparação de métodos para conjuntos de dados Qwen2-1.5 com diferentes categorias de raciocínio.

Dataset	**Original**	**(Our) 2T-FT**
ARC Easy	33.11 (± 1.38)	**33.62** (± 1.38)
ARC Challenge	**66.71** (± 0.97)	66.67 (± 0.97)
LogiQA	**27.04** (± 1.74)	26.88 (± 1.74)
GPQA	**28.12** (± 2.13)	27.68 (± 2.12)
PIQA	**75.73** (± 1.00)	**75.73** (± 1.00)
OpenBookQA	26.60 (± 1.98)	**27.40** (± 2.00)
BBH Causal Judgement	**52.41** (± 3.66)	51.87 (± 3.66)
BBH Boolean Expressions	**58.40** (± 3.12)	40.40 (± 3.11)
ETHICS	62.40 (± 0.76)	**63.30** (± 0.33)
MUTUAL	**69.54** (± 1.03)	69.52 (± 1.02)

5 Conclusão

O presente trabalho apresentou a abordagem *Two-Token Fine-Tuning (2T-FT)* como uma alternativa eficiente e econômica para aprimorar a capacidade de raciocínio dos *Large Language Models (LLMs)*, reduzindo significativamente os custos computacionais associados ao método de *fine-tuning* tradicional. A proposta explora o uso de *Chain-of-Thought (CoT)* já implícitos nas respostas geradas pelo modelo, realizando o ajuste fino de apenas dois *tokens* – o primeiro *token* da resposta e o primeiro *token* de fim de sentença (EOS).

Os experimentos demonstraram que a abordagem *2T-FT* melhora substancialmente a geração de respostas contendo raciocínio passo a passo (CoT), sem a necessidade de modificar toda a estrutura do modelo ou recorrer a técnicas intensivas de *prompt engineering*.

Entre os pontos fortes do *2T-FT*, destaca-se a sua eficiência computacional, tornando o *fine-tuning* acessível mesmo para usuários com recursos limitados. Adicionalmente, a abordagem preserva a estrutura original do modelo, facilitando a implementação em diversos cenários. No entanto, a técnica também apresenta limitações. Por depender exclusivamente de dois *tokens*, sua eficácia pode variar entre diferentes tipos de tarefas, especialmente aquelas que exigem raciocínio mais complexo ou com múltiplas etapas interdependentes. Além disso, sua aplicação requer uma seleção cuidadosa dos *tokens*-alvo para garantir que o ajuste fino seja realmente benéfico.

Comparado ao *fine-tuning* tradicional, o *2T-FT* revelou-se uma solução promissora para aplicações em que o custo computacional é um fator crítico, principalmente visando treinamentos menos custoso e com poucos dados. Ademais, o método contribui para a discussão sobre a capacidade de raciocínio dos *LLMs*, destacando-se como um passo em direção ao desenvolvimento de modelos mais coerentes e precisos.

Como trabalho futuro, sugere-se explorar o *2T-FT* em tarefas além do raciocínio aritmético, avaliando sua aplicabilidade em problemas de lógica, inferência semântica e tomada de decisão. Além disso, a integração do método com outras estratégias além das utilizadas, como *self-consistency*, pode ampliar seu potencial, tornando a geração de respostas ainda mais precisa e robusta.

Conclui-se, portanto, que a técnica *Two-Token Fine-Tuning* oferece uma solução alternativa para aprimorar o raciocínio em *LLMs*, e enfatiza viabilidade de métodos de *fine-tuning* minimalistas, cominando em avanços significativos na eficiência dos modelos de linguagem, no menor consumo de dados nas etapas de treinamento e na simulação de raciocínio durante o tempo de inferência.

Referências

AINSLIE, J. et al. *GQA: Training Generalized Multi-Query Transformer Models from Multi-Head Checkpoints*. 2023. Citado na página 25.

ALLEN-ZHU, Z.; LI, Y. *Physics of Language Models: Part 3.2, Knowledge Manipulation*. 2023. Citado na página 33.

ANIL, R. et al. *PaLM 2 Technical Report*. 2023. Citado na página 26.

ANTHROPIC. *Claude 3 Haiku: Our Fastest Model Yet*. 2024. Disponível em: <https://www.anthropic.com/news/claude-3-haiku>. Citado na página 26.

BA, J. L.; KIROS, J. R.; HINTON, G. E. *Layer Normalization*. 2016. Citado na página 24.

BANSAL, H. et al. *Rethinking the Role of Scale for In-Context Learning: An Interpretability-based Case Study at 66 Billion Scale*. 2023. Citado na página 31.

BISK, Y. et al. *PIQA: Reasoning about Physical Commonsense in Natural Language*. 2019. Disponível em: <https://arxiv.org/abs/1911.11641>. Citado na página 43.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. Citado na página 17.

BROWN, T. B. et al. *Language Models are Few-Shot Learners*. 2020. Citado 2 vezes nas páginas 26 e 31.

CHALVATZAKI, G. et al. Learning to reason over scene graphs: a case study of finetuning gpt-2 into a robot language model for grounded task planning. *Frontiers in Robotics and AI*, v. 10, 08 2023. Citado na página 26.

CHEN, B. et al. *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. 2023. Citado na página 17.

CHEN, W. et al. *Program of Thoughts Prompting: Disentangling Computation from Reasoning for Numerical Reasoning Tasks*. 2023. Citado 3 vezes nas páginas 17, 18 e 32.

CHUANG, Y.-S. et al. *DoLa: Decoding by Contrasting Layers Improves Factuality in Large Language Models*. 2024. Citado na página 36.

CHUNG, H. W. et al. *Scaling Instruction-Finetuned Language Models*. 2022. Citado 2 vezes nas páginas 17 e 31.

COBBE, K. et al. *Training Verifiers to Solve Math Word Problems*. 2021. Citado 2 vezes nas páginas 17 e 43.

CUI, L. et al. MuTual: A dataset for multi-turn dialogue reasoning. In: JURAFSKY, D. et al. (Ed.). *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, 2020. p. 1406–1416. Disponível em: <https://aclanthology.org/2020.acl-main.130>. Citado na página 43.

DAO, T. *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*. 2023. Citado na página 27.

DETTMERS, T. et al. *QLoRA: Efficient Finetuning of Quantized LLMs*. 2023. Citado na página 30.

DEVLIN, J. et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 2019. Citado na página 25.

DING, J. et al. *LongNet: Scaling Transformers to 1,000,000,000 Tokens*. 2023. Citado na página 27.

DONG, Y.; CORDONNIER, J.-B.; LOUKAS, A. Attention is not all you need: pure attention loses rank doubly exponentially with depth. In: MEILA, M.; ZHANG, T. (Ed.). *Proceedings of the 38th International Conference on Machine Learning*. PMLR, 2021. (Proceedings of Machine Learning Research, v. 139), p. 2793–2803. Disponível em: <https://proceedings.mlr.press/v139/dong21a.html>. Citado na página 25.

DOSOVITSKIY, A. et al. *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. 2021. Citado na página 25.

FAN, A.; LEWIS, M.; DAUPHIN, Y. *Hierarchical Neural Story Generation*. 2018. Citado na página 35.

FU, Y. et al. *Complexity-Based Prompting for Multi-Step Reasoning*. 2023. Citado na página 17.

GAO, L. et al. *PAL: Program-aided Language Models*. 2023. Citado 3 vezes nas páginas 17, 18 e 32.

GAO, L. et al. *A framework for few-shot language model evaluation*. Zenodo, 2023. Disponível em: <https://zenodo.org/records/10256836>. Citado na página 43.

GOODFELLOW, I. J.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. <http://www.deeplearningbook.org>. Citado na página 22.

GRAVES, A. *Sequence Transduction with Recurrent Neural Networks*. 2012. Citado na página 34.

GUNASEKAR, S. et al. *Textbooks Are All You Need*. 2023. Citado na página 27.

GUNASEKAR, S. et al. Textbooks are all you need. *CoRR*, abs/2306.11644, 2023. Disponível em: <https://doi.org/10.48550/arXiv.2306.11644>. Citado na página 43.

HE, K. et al. *Deep Residual Learning for Image Recognition*. 2015. Citado na página 24.

HOCHREITER, S.; SCHMIDHUBER, J. Long short-term memory. *Neural Computation*, v. 9, n. 8, p. 1735–1780, 1997. Citado na página 22.

HOLTZMAN, A. et al. *The Curious Case of Neural Text Degeneration*. 2020. Citado na página 35.

HU, E. J. et al. *LoRA: Low-Rank Adaptation of Large Language Models*. 2021. Citado 2 vezes nas páginas 29 e 43.

HUANG, C. et al. *LoraHub: Efficient Cross-Task Generalization via Dynamic LoRA Composition*. 2024. Citado na página 29.

HUANG, D. et al. *CodeCoT: Tackling Code Syntax Errors in CoT Reasoning for Code Generation*. 2024. Citado na página 31.

HUANG, J.; CHANG, K. C.-C. *Towards Reasoning in Large Language Models: A Survey*. 2023. Citado na página 17.

HUANG, J. et al. *Large Language Models Cannot Self-Correct Reasoning Yet*. 2024. Citado na página 35.

Information is Beautiful. *The rise of generative AI: Large Language Models (LLMs) like ChatGPT*. 2024. <https://informationisbeautiful.net/visualizations/the-rise-of-generative-ai-large-language-models-llms-like-chatgpt/>. Accessed: 2024-12-25. Citado na página 27.

JIANG, A. Q. et al. *Mistral 7B*. 2023. Citado na página 17.

JIANG, A. Q. et al. *Mixtral of Experts*. 2024. Citado na página 27.

JOSHI, M. et al. *SpanBERT: Improving Pre-training by Representing and Predicting Spans*. 2020. Disponível em: <https://arxiv.org/abs/1907.10529>. Citado na página 38.

KAPLAN, J. et al. *Scaling Laws for Neural Language Models*. 2020. Citado na página 26.

KINGMA, D. P.; BA, J. *Adam: A Method for Stochastic Optimization*. 2017. Disponível em: <https://arxiv.org/abs/1412.6980>. Citado na página 25.

KOJIMA, T. et al. Large language models are zero-shot reasoners. In: OH, A. H. et al. (Ed.). *Advances in Neural Information Processing Systems*. [s.n.], 2022. Disponível em: <https://openreview.net/forum?id=e2TBb5y0yFf>. Citado na página 44.

KOJIMA, T. et al. *Large Language Models are Zero-Shot Reasoners*. 2023. Citado 4 vezes nas páginas 32, 40, 41 e 43.

KUPTSOV, P. V.; KUPTSOVA, A. V.; STANKEVICH, N. V. Artificial neural network as a universal model of nonlinear dynamical systems. *Nelineinaya Dinamika*, Izhevsk Institute of Computer Science, v. 17, n. 1, p. 5–21, 2021. ISSN 2658-5316. Disponível em: <http://dx.doi.org/10.20537/nd210102>. Citado na página 21.

KWON, W. et al. Efficient memory management for large language model serving with pagedattention. In: *Proceedings of the 29th Symposium on Operating Systems Principles*. New York, NY, USA: Association for Computing Machinery, 2023. (SOSP '23), p. 611–626. ISBN 9798400702297. Disponível em: <https://doi.org/10.1145/3600006.3613165>. Citado na página 44.

LECUN, Y. et al. Gradient-based learning applied to document recognition. In: *Proceedings of the IEEE*. [s.n.], 1998. v. 86, n. 11, p. 2278–2324. Disponível em: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.42.7665>. Citado na página 22.

LEWIS, M. et al. *BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension*. 2019. Citado na página 26.

LEWIS, P. et al. *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*. 2021. Citado na página 28.

LI, J. et al. *Structured Chain-of-Thought Prompting for Code Generation*. 2023. Citado na página 31.

LIALIN, V. et al. *Scaling Down to Scale Up: A Guide to Parameter-Efficient Fine-Tuning*. 2024. Disponível em: <https://arxiv.org/abs/2303.15647>. Citado na página 29.

LIN, Z. et al. *Rho-1: Not All Tokens Are What You Need*. 2024. Disponível em: <https://arxiv.org/abs/2404.07965>. Citado 3 vezes nas páginas 18, 41 e 46.

LING, W. et al. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In: BARZILAY, R.; KAN, M.-Y. (Ed.). *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, 2017. p. 158–167. Disponível em: <https://aclanthology.org/P17-1015>. Citado na página 31.

LIU, J. et al. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. In: BESSIERE, C. (Ed.). *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*. International Joint Conferences on Artificial Intelligence Organization, 2020. p. 3622–3628. Main track. Disponível em: <https://doi.org/10.24963/ijcai.2020/501>. Citado na página 43.

LIU, S.-Y. et al. *DoRA: Weight-Decomposed Low-Rank Adaptation*. 2024. Citado na página 29.

MA, S. et al. *The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits*. 2024. Citado na página 27.

MADAAN, A. et al. *Language Models of Code are Few-Shot Commonsense Learners*. 2022. Citado na página 33.

MEISTER, C. et al. *Locally Typical Sampling*. 2023. Citado na página 36.

MELZ, E. *Enhancing LLM Intelligence with ARM-RAG: Auxiliary Rationale Memory for Retrieval Augmented Generation*. 2023. Citado na página 28.

MERITY, S. et al. Pointer sentinel mixture models. In: *International Conference on Learning Representations*. [s.n.], 2017. Disponível em: <https://openreview.net/forum?id=Byj72udxe>. Citado na página 43.

MIHAYLOV, T. et al. Can a suit of armor conduct electricity? a new dataset for open book question answering. In: RILOFF, E. et al. (Ed.). *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018. p. 2381–2391. Disponível em: <https://aclanthology.org/D18-1260>. Citado na página 43.

MITRA, A. et al. *Orca 2: Teaching Small Language Models How to Reason*. 2023. Citado na página 27.

MOSBACH, M. et al. *Few-shot Fine-tuning vs. In-context Learning: A Fair Comparison and Evaluation*. 2023. Citado na página 30.

NYE, M. et al. *Show Your Work: Scratchpads for Intermediate Computation with Language Models*. 2021. Citado 2 vezes nas páginas 17 e 32.

O'BRIEN, S.; LEWIS, M. *Contrastive Decoding Improves Reasoning in Large Language Models*. 2023. Citado na página 34.

OLSSON, C. et al. *In-context Learning and Induction Heads*. 2022. Citado na página 31.

OUYANG, L. et al. *Training language models to follow instructions with human feedback*. 2022. Citado na página 28.

PATEL, A.; BHATTAMISHRA, S.; GOYAL, N. *Are NLP Models really able to Solve Simple Math Word Problems?* 2021. Citado na página 43.

PETERS, M. E. et al. *Deep contextualized word representations*. 2018. Citado na página 26.

POPE, R. et al. Efficiently scaling transformer inference. *CoRR*, abs/2211.05102, 2022. Disponível em: <https://doi.org/10.48550/arXiv.2211.05102>. Citado na página 25.

QIAO, S. et al. *Reasoning with Language Model Prompting: A Survey*. 2023. Citado na página 17.

RADFORD, A. et al. Improving language understanding by generative pre-training. 2018. Citado na página 25.

RAJPURKAR, P. et al. *SQuAD: 100,000+ Questions for Machine Comprehension of Text*. 2016. Citado 2 vezes nas páginas 28 e 38.

REIN, D. et al. GPQA: A graduate-level google-proof q&a benchmark. In: *First Conference on Language Modeling*. [s.n.], 2024. Disponível em: <https://openreview.net/forum?id=Ti67584b98>. Citado na página 43.

RENZE, M.; GUVEN, E. *The Effect of Sampling Temperature on Problem Solving in Large Language Models*. 2024. Disponível em: <https://arxiv.org/abs/2402.05201>. Citado na página 35.

ROY, S.; ROTH, D. Solving general arithmetic word problems. In: MÀRQUEZ, L.; CALLISON-BURCH, C.; SU, J. (Ed.). *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 1743–1752. Disponível em: <https://aclanthology.org/D15-1202>. Citado na página 43.

RUSSELL, S.; NORVIG, P. *Artificial Intelligence: A Modern Approach*. 3. ed. [S.l.]: Prentice Hall, 2010. Citado na página 21.

SAHOO, P. et al. *A Systematic Survey of Prompt Engineering in Large Language Models: Techniques and Applications*. 2024. Citado na página 17.

SANH, V. et al. *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. 2020. Citado na página 27.

SEGAL, E. et al. *A Simple and Effective Model for Answering Multi-span Questions*. 2020. Disponível em: <https://arxiv.org/abs/1909.13375>. Citado na página 38.

SHAZEER, N. *Fast Transformer Decoding: One Write-Head is All You Need*. 2019. Citado na página 25.

SHI, C. et al. *A Thorough Examination of Decoding Methods in the Era of LLMs*. 2024. Citado na página 36.

SU, J. et al. *RoFormer: Enhanced Transformer with Rotary Position Embedding*. 2023. Citado na página 25.

SUTSKEVER, I.; VINYALS, O.; LE, Q. V. *Sequence to Sequence Learning with Neural Networks*. 2014. Citado na página 34.

SUZGUN, M. et al. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In: ROGERS, A.; BOYD-GRABER, J.; OKAZAKI, N. (Ed.). *Findings of the Association for Computational Linguistics: ACL 2023*. Toronto, Canada: Association for Computational Linguistics, 2023. p. 13003–13051. Disponível em: <https://aclanthology.org/2023.findings-acl.824>. Citado na página 43.

TEAM, G. et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. Citado na página 26.

TIAN, Y. et al. *TinyLLM: Learning a Small Student from Multiple Large Language Models*. 2024. Citado na página 27.

TOUVRON, H. et al. *LLaMA: Open and Efficient Foundation Language Models*. 2023. Citado na página 17.

VALIPOUR, M. et al. *DyLoRA: Parameter Efficient Tuning of Pre-trained Models using Dynamic Search-Free Low-Rank Adaptation*. 2023. Citado na página 29.

VASWANI, A. et al. *Attention Is All You Need*. 2017. Citado 2 vezes nas páginas 22 e 24.

VERMA, P.; BERGER, J. *Audio Transformers:Transformer Architectures For Large Scale Audio Understanding. Adieu Convolutions*. 2021. Citado na página 25.

WANG, B. et al. *Towards Understanding Chain-of-Thought Prompting: An Empirical Study of What Matters*. 2023. Citado na página 17.

WANG, T. et al. What language model architecture and pretraining objective works best for zero-shot generalization? In: CHAUDHURI, K. et al. (Ed.). *Proceedings of the 39th International Conference on Machine Learning*. PMLR, 2022. (Proceedings of Machine Learning Research, v. 162), p. 22964–22984. Disponível em: <https://proceedings.mlr.press/v162/wang22u.html>. Citado na página 25.

WANG, X. et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. 2023. Citado 2 vezes nas páginas 35 e 37.

WANG, X.; ZHOU, D. *Chain-of-Thought Reasoning Without Prompting*. 2024. Citado na página 18.

WANG, X.; ZHOU, D. *Chain-of-Thought Reasoning Without Prompting*. 2024. Disponível em: <https://arxiv.org/abs/2402.10200>. Citado 2 vezes nas páginas 18 e 44.

WANG, X.; ZHOU, D. *Chain-of-Thought Reasoning Without Prompting*. 2024. Citado 2 vezes nas páginas 34 e 35.

WANG, Z. et al. *StyleAdapter: A Single-Pass LoRA-Free Model for Stylized Image Generation*. 2023. Citado na página 30.

WEI, J. et al. Finetuned language models are zero-shot learners. In: *International Conference on Learning Representations*. [s.n.], 2022. Disponível em: <https://openreview.net/forum?id=gEZrGCozdqR>. Citado na página 18.

WEI, J. et al. *Finetuned Language Models Are Zero-Shot Learners*. 2022. Citado 2 vezes nas páginas 17 e 31.

WEI, J. et al. *Emergent Abilities of Large Language Models*. 2022. Citado na página 31.

WEI, J. et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. Citado 3 vezes nas páginas 17, 31 e 32.

WU, Z. et al. *ReFT: Representation Finetuning for Language Models*. 2024. Citado na página 30.

XIE, S. M. et al. *An Explanation of In-context Learning as Implicit Bayesian Inference*. 2022. Citado na página 17.

XIONG, R. et al. On layer normalization in the transformer architecture. In: III, H. D.; SINGH, A. (Ed.). *Proceedings of the 37th International Conference on Machine Learning*. PMLR, 2020. (Proceedings of Machine Learning Research, v. 119), p. 10524–10533. Disponível em: <https://proceedings.mlr.press/v119/xiong20b.html>. Citado na página 25.

XU, L. et al. *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*. 2023. Citado 2 vezes nas páginas 28 e 29.

XU, Y. et al. *QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models*. 2023. Citado na página 30.

YADAV, V.; BETHARD, S.; SURDEANU, M. Quick and (not so) dirty: Unsupervised selection of justification sentences for multi-hop question answering. In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. Disponível em: <http://dx.doi.org/10.18653/v1/D19-1260>. Citado na página 43.

YANG, A. et al. *Qwen2 Technical Report*. 2024. Disponível em: <https://arxiv.org/abs/2407.10671>. Citado na página 43.

YANG, Y.; YIH, W.-t.; MEEK, C. WikiQA: A challenge dataset for open-domain question answering. In: MÀRQUEZ, L.; CALLISON-BURCH, C.; SU, J. (Ed.). *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Lisbon, Portugal: Association for Computational Linguistics, 2015. p. 2013–2018. Disponível em: <https://aclanthology.org/D15-1237>. Citado na página 28.

YAO, Y. et al. *Learning From Correctness Without Prompting Makes LLM Efficient Reasoner*. 2024. Citado na página 35.