UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

CAIO CEZAR QUEIROZ

DESENVOLVIMENTO DE UM *FRAMEWORK* DE *BENCHMARKING* PARA AVALIAÇÃO DE MODELOS DE LINGUAGEM PROPRIETÁRIOS E *OPEN SOURCE*

UBERLÂNDIA
2026

CAIO CEZAR QUEIROZ

DESENVOLVIMENTO DE UM *FRAMEWORK* DE *BENCHMARKING* PARA AVALIAÇÃO DE MODELOS DE LINGUAGEM PROPRIETÁRIOS E *OPEN SOURCE*

Trabalho apresentado como requisito de aprovação na disciplina Trabalho de Conclusão de Curso, do Curso de Engenharia de Computação da Universidade Federal de Uberlândia como requisito para obtenção do título de Bacharel em Engenharia de Computação.

Orientador: Prof. Dr. Kil Jin Brandini Park

UBERLÂNDIA

2026

CAIO CEZAR QUEIROZ

Trabalho apresentado como requisito de aprovação na disciplina Trabalho de Conclusão de Curso, do Curso de Engenharia de Computação da Universidade Federal de Uberlândia como requisito para obtenção do título de Bacharel em Engenharia de Computação.

Uberlândia, 07 de julho de 2026

Banca Examinadora:

_____

Prof. Dr. Kil Jin Brandini Park
Orientador

_____

Prof. Dr. Alexandre Cardoso

_____

Prof. Me. Lucas Gonçalves Cunha

AGRADECIMENTOS

Agradeço, primeiramente, aos meu pais Jeovanny Queiroz e Renata Aparecida Saldanha Queiroz pelo apoio incondicional, incentivo e por serem os pilares fundamentais para minha formação pessoal e profissional.

Aos amigos e colegas, pelo companheirismo, apoio e troca de experiências ao longo dessa jornada. A todo corpo docente e técnico da Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia pelo apoio durante a graduação.

Por fim, agradeço a todos que, direta ou indiretamente, contribuíram para a realização deste trabalho.

“Os limites da minha linguagem significam os limites do meu mundo.”

Ludwig Wittgenstein

RESUMO

As tecnologias de inteligência artificial generativa, em especial os grandes modelos de linguagem (*Large Language Models* – LLMs), têm se destacado como ferramentas centrais em diversas aplicações computacionais. Entretanto, a predominância de soluções proprietárias levanta questões relacionadas à transparência, flexibilidade e dependência tecnológica. Assim, o objetivo deste trabalho é apresentar e avaliar um *framework* experimental de comparação entre modelos de linguagem proprietários e *open source*, utilizando métricas quantitativas e *benchmarks* padronizados. Foram empregados modelos da família Gemini (*Google*) e alternativas *open source*, como *LLaMA*, *GPT-OSS e Qwen*, avaliados por meio da *Google Generative AI API* e da *Groq API*, respectivamente. O estudo adota uma abordagem quantitativa e experimental, utilizando as métricas *ROUGE*, *BLEU*, *BERTScore* e os *benchmarks MMLU* e *HellaSwag*, complementados por análises com a ferramenta *EvidentlyAI*. Os resultados demonstram a viabilidade do sistema como ferramenta de *benchmarking*, ressaltando o potencial das soluções abertas e a importância de metodologias transparentes para a avaliação de LLMs.

Palavras-chave: Inteligência Artificial Generativa; LLM; *Open Source*; Grandes Modelos de Linguagem; *Benchmarking*.

ABSTRACT

Generative artificial intelligence technologies, especially *Large Language Models* (LLMs), have emerged as central tools in computational applications. However, the predominance of proprietary solutions raises questions about transparency, flexibility, and technological dependency. Therefore, the objective of this work is to present and evaluate an experimental *framework* for comparing proprietary and *open-source* language models, using quantitative metrics and standardized *benchmarks*. Models from the Gemini family (Google) and *open-source* alternatives such as *LLaMA*, *GPT-OSS and Qwen* were employed, evaluated through the *Google Generative AI API* and *Groq API*, respectively. The study adopts a quantitative and experimental approach, using the *ROUGE*, *BLEU*, and *BERTScore* metrics, as well as the *MMLU* and *HellaSwag benchmarks*, complemented by analyses with the *EvidentlyAI* tool. The results demonstrate the feasibility of the system as a *benchmarking* tool, highlighting the potential of open solutions and the importance of transparent methodologies for LLM evaluation.

Keywords: Generative Artificial Intelligence; LLM; *Open Source*; *Large Language Models*; *Benchmarking*.

LISTA DE ABREVIATURAS

AI	Artificial Intelligence
API	Application Programming Interface
BLEU	Bilingual Evaluation Understudy
BPE	Byte Pair Encoding
CSV	Comma-Separated Values
GB	Gigabytes
GPU	Graphics Processing Unit
GPT	Generative Pre-trained Transformer
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IA	Inteligência Artificial
ICLR	International Conference on Learning Representations
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
LLM	Large Language Model
MLA	Multi-Head Latent Attention
MLP	Multi Layer Perceptrons
MMLU	Massive Multitask Language Understanding
MoE	Mixture of Experts
OSI	Open Source Initiative
PoC	Proof of Concept
PLN	Processamento de Linguagem Natural
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
RNA	Redes Neurais Artificiais
SVM	Support Vector Machine
TB	Terabytes

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Embora o conceito de inteligência artificial (IA) remonte à década de 1940, quando Alan Turing propôs o chamado “jogo da imitação”, segundo o qual as máquinas poderiam “pensar” como humanos, foi apenas com os avanços recentes em *hardware*, *big data* e processamento em nuvem que se tornou possível desenvolver sistemas que se aproximam da ideia de máquinas capazes de aprender e raciocinar (TURING, 1950).

Durante esse processo, a IA também passou por uma evolução, incorporando algoritmos baseados em redes neurais, *machine learning*, *deep learning*, processamento de linguagem natural e, mais recentemente, redes generativas. Estas, fundamentadas em arquiteturas neurais competitivas (uma rede contra a outra), passaram a produzir imagens e textos de altíssima qualidade (RAMOS, 2023).

Esse avanço abriu espaço para os grandes modelos de linguagem, os *Large Language Models* (LLMs), que se destacam pela capacidade de geração e compreensão de texto em larga escala. Um dos principais marcos foi o lançamento do Chat GPT em 2022, pela OpenAI, treinado com mais de 570 GB de dados, um acervo equivalente a mais de 40 bilhões de palavras, e composto por modelos com mais de 175 bilhões de parâmetros (SANTOS, 2024).

Com essa popularização, a IA generativa torna-se mais acessível ao usuário, segundo Röhe e Santaella (2023), uma vez que sua interação ocorre por meio de entradas simples de texto, conhecidas como *prompts*. Além disso, quanto mais detalhados esses comandos, melhor tende a ser o resultado obtido, sem a necessidade de conhecimento em programação ou acesso ao código-fonte dos sistemas.

Além disso, mesmo que modelos proprietários, como o Chat GPT (OpenAI), o Gemini (Google) e o Claude (Anthropic) estejam amplamente difundidos, alternativas *open source*, como o LLaMA (Meta), GPT-OSS e o Qwen (Alibaba), ainda recebem pouca atenção. Por serem abertos, esses modelos oferecem maior transparência, flexibilidade e possibilidade de auditoria pela comunidade, fatores essenciais para reduzir a dependência tecnológica.

Nesse contexto, o desenvolvimento de um *framework* de *benchmarking* permite estabelecer uma base comparativa estruturada, fornecendo não apenas resultados pontuais, mas também uma metodologia passível de replicação e aprimoramento em estudos futuros. Portanto, este trabalho se justifica pela necessidade de criar ferramentas objetivas de comparação que subsidiem decisões técnicas e estratégicas no uso de LLMs, tanto em contextos acadêmicos quanto corporativos.

Assim, a presente pesquisa tem como propósito desenvolver e validar um *framework* de *benchmarking* automatizado entre modelos de linguagem proprietários e *open source*, utilizando métricas automáticas, *benchmarks* padronizados e ferramentas de análise quantitativa.

Para alcançar esse objetivo, o estudo propõe-se implementar uma arquitetura de software modular, assegurando a possibilidade de integração de novas APIs e a extensibilidade para diferentes *datasets* de teste. Busca-se ainda integrar ferramentas de testes padronizados, como MMLU e HellaSwag, e aplicar métricas de avaliação automática, como ROUGE, BLEU e BERTScore, em conjunto com análises estatísticas no EvidentlyAI, a fim de comparar e realizar uma Prova de Conceito (PoC) que valide o funcionamento do *framework*, identificando limitações técnicas e propondo melhorias futuras.

Para embasar essa investigação e compreender as tecnologias subjacentes ao *framework,* o próximo capítulo busca fundamentar o conceito de Inteligência Artificial Generativa, abordando seu funcionamento por meio de Redes Neurais, *Deep Learning*, arquitetura *Transformers* e Processamento de Linguagem Natural (PLN). Além disso, pretende-se diferenciar os paradigmas de modelos de código fechado (proprietários), código aberto (*open source*) e pesos abertos (*open weights*), discutindo suas importâncias no contexto atual das IAs generativas e descrevendo as principais famílias de modelos utilizadas, com destaque para suas arquiteturas, escalabilidade e características técnicas.

2 REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta a base teórica que fundamenta o trabalho, abordando conceitos essenciais como aprendizado de máquina, processamento de linguagem natural, inteligência artificial generativa e código aberto. A fundamentação teórica tem como objetivo sustentar a construção do *framework* experimental, evidenciando como os avanços em *deep learning* e *transformers* viabilizaram o desenvolvimento dos LLMs modernos, bem como o papel do movimento *open source* no estímulo ao desenvolvimento colaborativo dessas tecnologias.

2.1 APRENDIZADO DE MÁQUINA

De acordo com Marsland (2011), o aprendizado de máquina pode ser definido como um campo da inteligência artificial dedicado ao estudo de algoritmos capazes de adquirir, de forma eficaz, a capacidade de aprender e resolver problemas de maneira autônoma. Em uma perspectiva mais voltada à engenharia, Mitchell (1997) apresenta a seguinte definição: “Diz-se que um programa de computador aprende pela experiência E, com respeito a algum tipo de tarefa T medida de performance P, se sua performance nas tarefas em T, conforme medida por P, melhora com a experiência E.”

Taulli (2020) define o aprendizado de máquina como um método computacional que utiliza a experiência para possibilitar o aprendizado, com o objetivo de realizar previsões ou otimizar o desempenho por meio de algoritmos. Nesse contexto, não é necessário que a máquina disponha previamente de uma fórmula matemática que relacione entradas e saídas, pois, a partir das informações fornecidas, muitas vezes por usuários ou outros programas, são geradas regras próprias que orientam a dedução dos resultados para determinados pontos amostrais.

É possível, ainda, traçar um paralelo entre o aprendizado de máquina e o aprendizado humano. Como exemplo, pode-se considerar uma pessoa que está aprendendo a dirigir: quando o instrutor solicita a redução da marcha do veículo, ocorre o aprendizado de uma prescrição do tipo condição-ação, relacionada a como e quando realizar essa mudança (MUELLER; MASSARON, 2019).

Géron (2021) afirma que não é necessário conhecer previamente o caminho entre a entrada e a saída, uma vez que os dados constituem as experiências mínimas necessárias para que as máquinas aprendam qual é a melhor resposta. Além disso, o aprendizado de máquina deve ser capaz de aprimorar continuamente sua base de conhecimento a cada processo de tomada de decisão, permitindo a aquisição constante de novas experiências (DOMINGOS, 2017).

Por fim, de acordo com Sugiyama et al. (2022), há diversas etapas que precisam ser executadas para viabilizar o aprendizado de máquina, como a coleta, a preparação e a verificação dos dados, com o objetivo de assegurar que as predições sejam, de fato, realizadas, conforme ilustrado na Figura 1.

Figura 1 — Passos para a execução do aprendizado de máquina

Fonte: Géron (2021).

Sendo assim, é importante destacar que todos os passos apresentados na figura anterior ocorrem de maneira iterativa, isto é, se um passo não é realizado como projetado, retorna-se à fase anterior para executar um ajuste. Além disso, somente após todos os passos que os algoritmos terão a capacidade de executar as previsões conforme as informações do mundo real. Recomenda-se, ainda, utilizar um passo de monitoramento a fim de avaliar se o modelo ainda executa boas previsões (MUELLER; MASSARON, 2019).

2.1.1 Tipos de Aprendizado de Máquina

De acordo com Mohammed, Khan e Bashier (2016), existem quatro categorias principais em que se pode separar as principais técnicas distintas de aprendizado, sendo elas: supervisionado, não supervisionado, semi-supervisionado e por reforço.

Os algoritmos do tipo supervisionados fazem o uso da experiência para obter conhecimento. Nesse caso, esses algoritmos são devidamente treinados por meio de características e rótulos definidos previamente, o que lhes permite reconhecer e classificar os dados segundo o aprendizado que foi adquirido na etapa de treinamento. Isto pode ser exemplificado na Figura 2, na qual ocorre a estratificação dos elementos em duas categorias distintas, de acordo com a cor e a forma (MOTTA, 2014).

Figura 2 — Aprendizado supervisionado

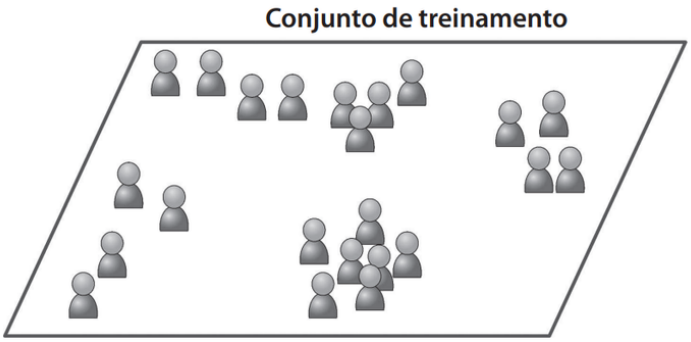
Fonte: Nielsen (2021).

De acordo com Alles (2018), ainda no que se refere ao aprendizado supervisionado é possível estratificá-los segundo os quais culminam em valores contínuos, os de regressão, os que dividem os dados em categorias e os de classificação. Entre os exemplos mais comuns de algoritmos supervisionados, pode-se mencionar: Naïve Bayes, k-vizinhos mais próximos, árvore de decisão e o SVM (máquina do vetor do suporte). Já os algoritmos não supervisionados são aqueles empregados quando não existem rótulos prévios para promover a classificação das informações. Assim, o processamento das informações da entrada se dá com o intuito de propiciar a criação de uma versão compacta ou um resumo dos dados (VEIGA; 2013).

Campos (2015) afirma que os algoritmos do tipo não supervisionado utilizam dados como entrada, mas não contam com uma resposta de saída predefinida. Como resultado, seu uso leva à formação de clusters (grupos) de dados que compartilham características em comum. A partir disso, torna-se necessária uma análise posterior para que se obtenha conhecimento sobre cada um desses agrupamentos. Entre os exemplos de algoritmos não supervisionados, destacam-se as redes neurais, os modelos probabilísticos e aqueles baseados em centroides.

A Figura 3 ilustra um conjunto de dados de treinamento empregado em aprendizado não supervisionado, no qual os dados não recebem rotulação. Nesse caso, eles são apenas organizados em grupos com base na similaridade de suas características, cabendo ao pesquisador atribuir a nomenclatura a cada conjunto.

Figura 3 — Aprendizado não supervisionado



Fonte: Géron (2021).

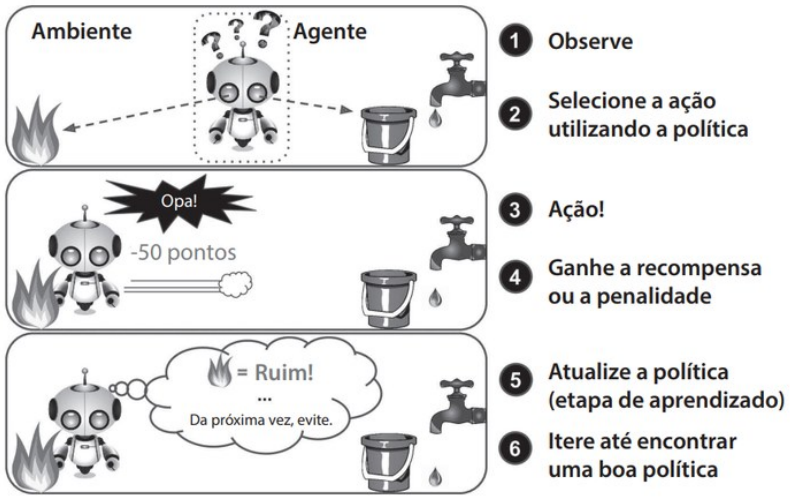
No que se refere aos algoritmos semi-supervisionados, observa-se um caráter intermediário em relação aos métodos anteriores. Sua principal característica, conforme ilustrado na Figura 4, é a capacidade de aprender a partir da combinação de dados rotulados e não rotulados (MOHAMED; KHAN; BASHIER, 2016).

Figura 4 — Aprendizado semi-supervisionado

Fonte: Gabriel (2022).

Por fim, destacam-se os algoritmos de aprendizado por reforço, que consistem em um cenário no qual um agente observa o ambiente, toma decisões e executa ações, recebendo como retorno recompensas ou penalidades. Essa interação permite que o agente aprenda, de forma autônoma, qual estratégia, denominada política, maximiza suas recompensas ao longo do tempo, orientando suas escolhas de ação em situações específicas. Um exemplo relevante de aplicação desse tipo de aprendizado é o programa *AlphaGo* da *DeepMind*, que em 2017 venceu o campeão mundial Ke Jie no jogo Go, como ilustrado na figura a seguir (Géron, 2021).

Figura 5 — Aprendizado por reforço

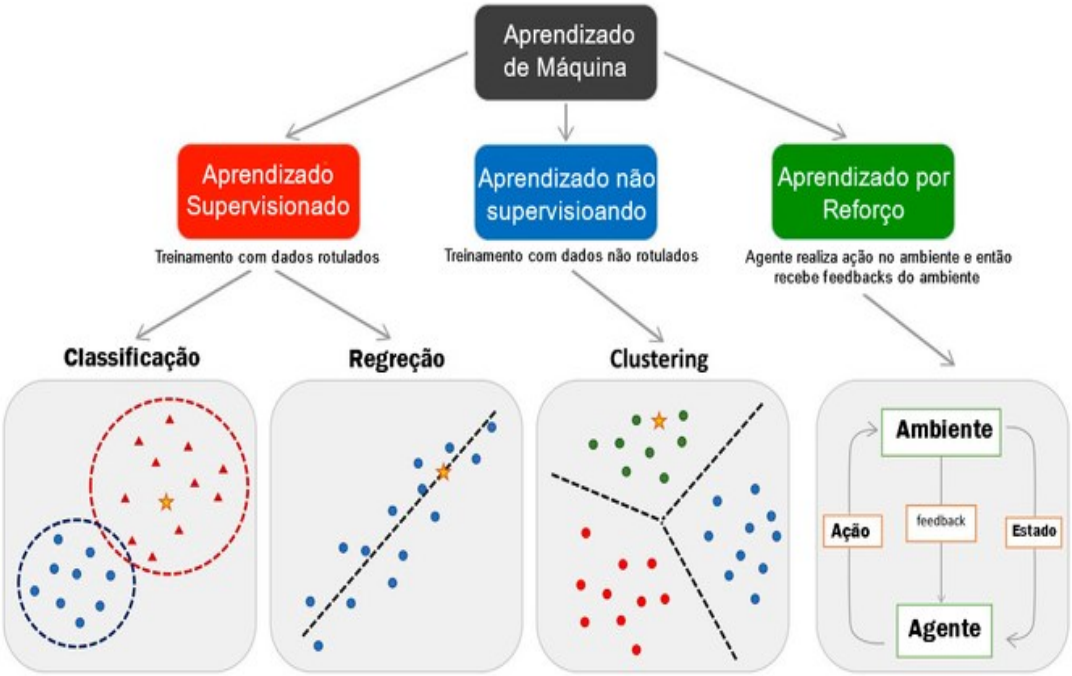


Fonte: Géron (2021).

Assim, cada um dos métodos de aprendizado de máquina apresenta particularidades próprias, tanto em relação à forma de processamento dos dados quanto aos tipos de problemas aos quais melhor se aplicam. Nesse sentido, a escolha do método mais adequado à finalidade pretendida mostra-se uma etapa fundamental, uma vez que impacta diretamente a capacidade do modelo de extrair padrões relevantes e produzir resultados consistentes. A correta adequação entre técnica e objetivo possibilita, portanto, o desenvolvimento de sistemas mais eficientes e alinhados às demandas estabelecidas no projeto.

Por fim, a Figura 6 apresenta uma sistematização dos principais tipos de aprendizado de máquina e de suas respectivas características, conforme proposto por Dambros (2019). Tal síntese contribui para uma compreensão integrada das abordagens discutidas, ao oferecer uma visão geral que evidencia suas diferenças, aplicações e especificidades.

Figura 6 — Os principais tipos de aprendizado de máquina



Fonte: Peng *et al.* (2021).

2.1.2 Redes Neurais e Aprendizado Profundo

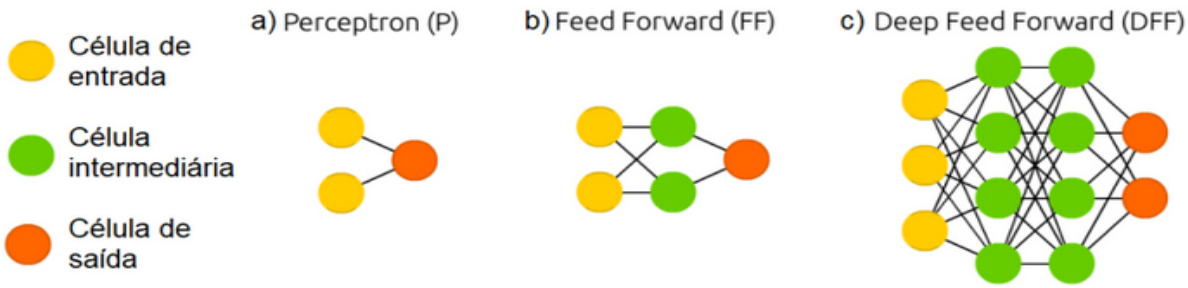
As redes neurais artificiais (RNA) constituem um campo do aprendizado de máquina bioinspirado no sistema nervoso dos animais, conforme apontam Basheer e Hajmeer (2000). Sua denominação decorre da semelhança entre a estrutura dessas redes e a organização dos neurônios presentes no cérebro humano, bem como de suas conexões. Pode-se afirmar que as redes neurais tiveram origem na década de 1940, quando McCulloch e Pitts (1943), inspirados no funcionamento do cérebro, desenvolveram um modelo matemático de rede neural. A partir desse marco inicial, observou-se um avanço significativo em sua capacidade e em suas aplicações ao longo do século XX.

Atualmente, as RNA representam um dos ramos mais amplos e avançados da inteligência artificial, alcançando resultados expressivos na busca por desenvolver máquinas com inteligência semelhante à humana, conforme destacam Iba e Noman (2020). Com o desenvolvimento tecnológico contemporâneo, observa-se a existência de uma ampla variedade de arquiteturas de redes neurais. No entanto, seus elementos centrais permanecem essencialmente os mesmos.

De acordo com Silvério (2015), de forma geral, essas redes são compostas por camadas de entrada, saída e camadas intermediárias, também denominadas, respectivamente, de *input*, *output* e *hidden layers*. Segundo Silva (2016), a camada de entrada corresponde ao estágio em

que os dados são inseridos na rede. A camada intermediária, por sua vez, é responsável pelo ajuste dos pesos sinápticos. Por fim, a camada de saída tem a função de apresentar os resultados finais do processamento da rede neural, conforme ilustrado na Figura 7, que apresenta três diferentes configurações de redes.

Figura 7 — Principais estruturas de uma rede neural



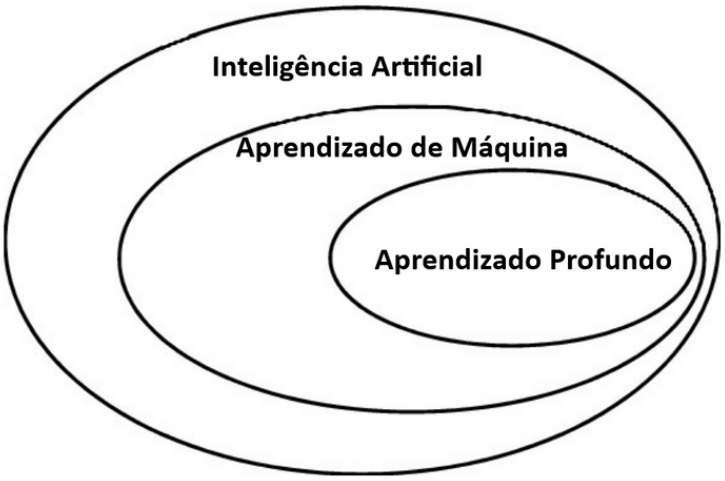
Fonte: Silva (2016).

De acordo com Haykin (2008), o primeiro modelo de rede neural, denominado *Perceptron*, consiste em um único neurônio artificial projetado para realizar classificações lineares ou binárias, sendo uma das estruturas mais simples existentes. A partir da combinação de vários neurônios, formam-se redes neurais de múltiplas camadas, conhecidas como *Multi Layer Perceptrons* (MLPs), nas quais cada neurônio de uma determinada camada está conectado a todos ou a um conjunto específico de neurônios da camada seguinte (IBA; NOMAN, 2020).

Nesse contexto, o aprendizado profundo (*deep learning*) tem como princípio explorar múltiplas camadas de processamento de informações para extrair e transformar características relevantes, tanto para aprendizado supervisionado quanto para não supervisionado. Esse conceito representa uma evolução natural dos *Perceptrons* multicamadas tradicionais, podendo envolver centenas ou até milhares de camadas, o que possibilita a aprendizagem de hierarquias complexas de representações (CICHY; KAISER, 2019).

Essa evolução das formas de inteligência artificial é ilustrada na Figura 8 por meio de um diagrama de Venn, evidenciando como, a cada avanço, os modelos se tornam mais precisos e “profundos”.

Figura 8 — Diagrama de Venn



Fonte: Adaptado de Mellit, Pavan e Ogliari (2020, p. 4).

Desse modo, o aprendizado profundo torna-se uma técnica adequada para problemas complexos como, por exemplo, reconhecimento de imagem e voz, processamento de linguagem natural, reconhecimento de padrões, tradução automática, entre outros (MELLIT; PAVAN; OGLIARI, 2020).

2.1.3 Transformers

Vaswani et al. (2017) propuseram a estrutura *Transformers*, originalmente projetada para computação paralela em GPUs, como um mecanismo de autoatenção cuja finalidade é capturar informações contextuais ao longo de sequências de forma mais eficiente do que abordagens baseadas em recorrência e conversão. Sua arquitetura foi inicialmente desenvolvida para tarefas de tradução automática e é composta por um codificador (*encoder*) e um decodificador (*decoder*). O codificador consiste em uma pilha de N = 6 camadas idênticas do *Transformer*, sendo que cada camada possui duas subcamadas.

A primeira subcamada corresponde a um mecanismo de autoatenção multi-*head*, enquanto a segunda consiste em uma *feed-forward* simples, totalmente conectada em posição rede. O decodificador, por sua vez, também é formado por uma pilha de 6 camadas e, além das duas subcamadas presentes em cada camada do codificador, inclui uma terceira subcamada responsável por realizar atenção sobre a saída do codificador.

A função de atenção pode ser descrita como um mapeamento entre uma consulta e um conjunto de pares chave-valor para gerar uma saída, sendo que consulta, chaves, valores e saída são todos representados por vetores. A saída é calculada como uma soma ponderada dos valores, em que os pesos são determinados por uma função de compatibilidade entre a consulta

e as chaves correspondentes. Em vez de aplicar uma única função de atenção com chaves, valores e consultas de dimensão *dmodel*, projeta-se linearmente esses elementos em múltiplas representações (h), com diferentes projeções aprendidas para as dimensões d_k, d_k e d_v, respectivamente.

Por fim, incorpora-se uma codificação posicional ao modelo, com o objetivo de introduzir informações sobre a posição relativa ou absoluta dos *tokens* na sequência.

2.2 IA GENERATIVA

Conforme Ramos (2023), a inteligência artificial generativa mais comum atualmente é o Chat GPT, desenvolvido pela OpenAI e fundamentado nos grandes modelos de linguagem (Large Language Models — LLMs). As ferramentas derivadas desse modelo apoiam atividades que vão desde a concepção da pesquisa, passando pela revisão de literatura e análise de dados, até a escrita e publicação de artigos científicos.

De acordo com Opara et al. (2023), entre os principais recursos proporcionados por essa tecnologia, destaca-se o processamento de linguagem natural, que permite o reconhecimento de padrões e tendências de linguagem, cujos modelos de linguagem de IA podem ajudar os pesquisadores a analisar e compreender grandes volumes de dados de texto. Além disso, a geração de texto constitui um recurso relevante, uma vez que possibilita a produção de conteúdos estruturados, úteis em tarefas como tradução automática e resumo de documentos.

Outro aspecto importante refere-se a criação de bases de dados, na qual os modelos de linguagem podem gerar dados úteis para o treinamento de algoritmos de aprendizado de máquina, contribuindo para a melhoria de seu desempenho. Por fim, destaca-se a capacidade de geração de imagens e áudios a partir de entradas textuais, permitindo a criação de imagens, logotipos, vídeos, bem como a simulação de vozes, inclusive de artistas.

2.3 PROCESSAMENTO DE LINGUAGEM NATURAL

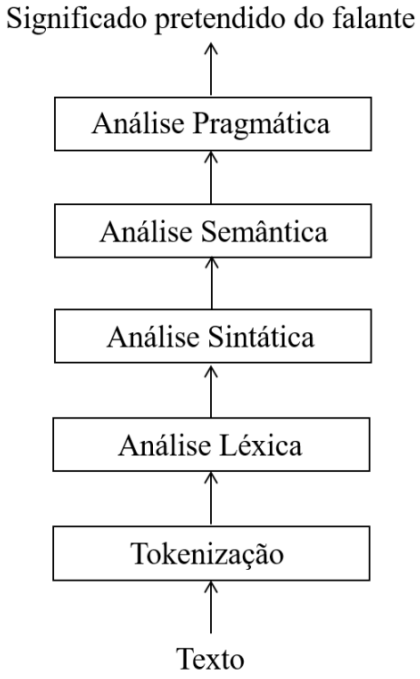
O Processamento de Linguagem Natural (PLN) pode ser definido como um conjunto de técnicas com objetivo de extrair o significado de textos escritos em linguagem natural e torná-los compreensíveis por computadores (CHOWDHURY, 2003). Os autores Bird, Klein e Loper (2009) definem linguagem natural como aquela utilizada por humanos em textos do cotidiano, como documentos e notícias, enquanto a linguagem artificial corresponde às notações matemáticas e às linguagens de programação. Pode-se afirmar que as linguagens naturais, ao

serem transmitidas de geração em geração, evoluem a ponto de tornar difícil a definição de regras explícitas.

De acordo com Indurkhya e Damerau (2010), a complexidade das linguagens naturais é abordada por meio de esforços voltados à identificação e compreensão dos conceitos linguísticos associados às palavras, como substantivos, adjetivos, verbos, entre outros. Essa abordagem busca analisar o texto não apenas como uma sequência de caracteres, mas também como uma estrutura hierárquica da linguagem. Dessa forma, implica a realização de análises léxicas, sintáticas, semânticas e morfológicas, a fim de compreender plenamente sua estrutura e significado.

As técnicas de PLN geralmente se iniciam com a separação das palavras, em um processo denominado "tokenização", cuja relação com as demais análises é apresentada na Figura 9, que ilustra as etapas de decomposição da linguagem natural.

Figura 9 — Etapas da análise no PLN



Fonte: Adaptado de Indurkhya e Damerau (2010, p. 4).

Sendo assim, o PLN pode ser compreendido como um campo interdisciplinar que integra várias áreas e dentre elas, a linguística computacional. Enquanto a linguagem é um objeto de estudo da linguística, no PLN o foco está no desenvolvimento de técnicas

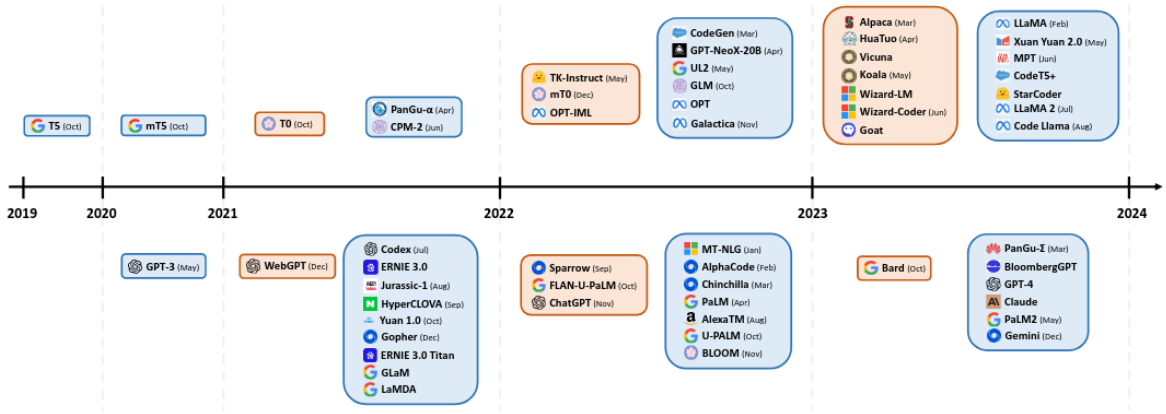
computacionais para ampliar a capacidade de processar e entender a linguagem humana natural, focando em tarefas como extração de informação de textos, tradução, respostas à perguntas e manter uma conversa (EISENSTEIN, 2019).

Sob o aspecto da engenharia, Chowdhury (2003) elucida que o processamento de linguagem natural tem como foco capacitar a linguagem humana no desenvolvimento de aplicações inovadoras para simplificar a interação entre computadores e linguagens humanas. Tendo isso em vista, pode-se destacar o aprendizado de máquinas como uma técnica que permite uma abordagem moderna para o processamento da linguagem natural.

2.3.1 Large Language Models (LLMs)

De acordo com Naveed et al. (2024), a linguagem desempenha um papel fundamental na facilitação da comunicação e da auto expressão humanas, especialmente na interação com máquinas. Nesse contexto, há uma crescente necessidade e demanda por modelos generalizados capazes de lidar com linguagens complexas, permitindo a realização de tarefas como criação de resumos, traduções e diversas formas de interação. Além disso, os autores destacam que, aliado ao aumento do poder de processamento e da capacidade de armazenamento de dados, tornou-se possível o desenvolvimento dos LLMs, os quais podem se aproximar do desempenho humano em diversas tarefas. A Figura 10 demonstra a evolução das técnicas de PLN até o surgimento dos LLMs atuais.

Figura 10 — Evolução histórica das técnicas de PLN até as LLMs



Fonte: Naveed et al. (2024).

Observa-se, então, que o PLN utiliza como base a modelagem estatística, enquanto as PLMs utilizam modelos de linguagem pré-treinados, tanto em ambientes supervisionados quanto não supervisionados, baseados em grande corpus textuais (DEVLIN et al., 2019). Tendo

em vista o maior desempenho das PLMs, houve uma evolução natural para os modelos LLMs, aumentando expressivamente os parâmetros dos modelos para a casa das centenas de milhares, bem como o conjunto de dados de treinamento para GB's e TB's (RAFFEL, 2020).

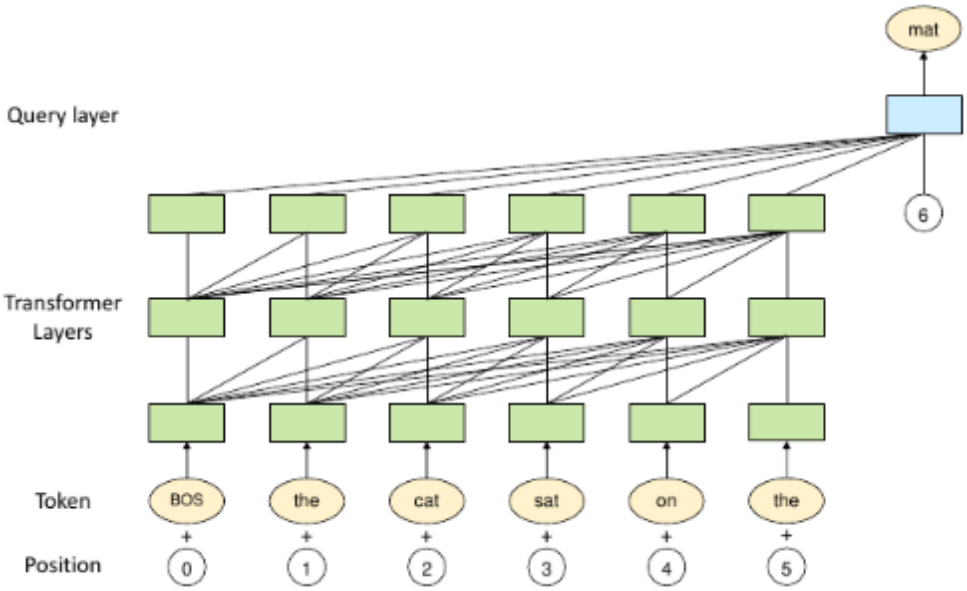
Segundo Webster (1992), o processo de tokenização é uma etapa essencial do pré-processamento no treinamento de LLMs. Os tokens podem ser caracteres, subpalavras, símbolos ou palavras, a depender do método de tokenização adotado, sendo os mais comuns o WordPiece, a codificação por par de bytes (BPE) e o UnigramLM. Após essa etapa, um transformador processa as sequências de entrada em paralelo e de forma independente, não capturando, por si só, informações de posição. Como resultado, codificações posicionais são introduzidas no modelo, por meio da adição de um vetor de incorporação posicional ao token, podendo assumir variações como codificação absoluta, relativa ou aprendida.

No contexto das codificações posicionais, Alibi e RoPE são amplamente utilizados. O Alibi subtrai um viés escalar da pontuação de atenção, o qual aumenta com a distância entre as posições dos tokens, favorecendo aqueles mais recentes. Já o RoPE rotaciona as representações de consultas e chaves em um ângulo proporcional à posição absoluta do token na sequência de entrada, resultando em um esquema de codificação posicional relativa que decai com a distância entre os tokens (NAVEED et al., 2024).

O mecanismo de atenção, por sua vez, atribui pesos aos tokens de entrada com base em sua relevância, permitindo que o modelo enfatize as informações mais importantes. Em transformadores, a atenção é calculada por meio dos mapeamentos de consulta, chave e valor das sequências de entrada. A pontuação de atenção é obtida a partir da multiplicação entre consulta e chave, sendo posteriormente utilizada para ponderar os valores.

Esse processo é ilustrado na Figura 11, na qual, de acordo com o token e o método de transformação utilizado, as palavras recebem diferentes pesos, refletindo a melhor organização possível na estrutura da frase. Ressalta-se que podem existir variações na arquitetura e no modelo adotado para a obtenção desses resultados.

Figura 11 — Exemplo de LLM



Fonte: Naveed et al. (2024).

2.3.2 Visão Geral dos Large Language Models

A evolução dos modelos de linguagens naturais tem avançado com grandes empresas tecnológicas liderando essa evolução. A Figura 12 resume os principais modelos de acordo com Naveed et al. (2024), detalhando características como o tipo de licença, o criador, o propósito, a quantidade de passos no treinamento, o processo utilizado na limpeza dos dados, a biblioteca utilizada, entre outros.

Embora grandes corporações como Google, Amazon e Microsoft dominem o cenário, observa-se a entrada de outros *players*, como Huawei, Bloomberg, Baidu, entre outros, que possuem potencial para impactar ainda mais o setor, com a criação e evolução das tecnologias em processamento de linguagem natural.

Figura 12 — Resumo dos principais modelos LLM

Models	Publication Venue	License Type	Model Creators	Purpose	No. of Params	Commercial Use	Steps Trained	Data/ Tokens	Data Cleaning	No. of Processing Units	Processing Unit Type	Training Time	Calculated Train. Cost	Training Parallelism	Library
T5 [10]	JMLR'20	Apache-2.0	Google	General	11B	✓	1M	1T	Heur+Dedup	1024	TPU v3	-	-	D+M	Mesh TensorFlow
GPT-3 [6]	NeurIPS'20	-	OpenAI	General	175B	×	-	300B	Dedup+QF	-	V100	-	-	M	-
mT5 [11]	NAACL'21	Apache-2.0	Google	General	13B	✓	1M	1T	-	-	-	-	-	-	-
PanGu-α [108]	arXiv'21	Apache-2.0	Huawei	General	200B	✓	260k	1.1TB	Heur+Dedup	2048	Ascend 910	-	-	D+OP+P+O+R	MindSpore
CPM-2 [12]	AI Open'21	MIT	Tsinghua	General	198B	✓	1M	2.6TB	Dedup	-	-	-	-	D+M	JAXFormer
Codex [131]	arXiv'21	-	OpenAI	Coding	12B	×	-	100B	Heur	-	-	-	-	-	-
ERNIE 3.0 [110]	arXiv'21	-	Baidu	General	10B	×	120k*	375B	Heur+Dedup	384	V100	-	-	M*	PaddlePaddle
Jurassic-1 [112]	White-Paper'21	Apache-2.0	AI21	General	178B	✓	-	300B	-	800	GPU	-	-	D+M+P	Megatron+DS
HyperCLOVA [114]	EMNLP'21	-	Naver	General	82B	×	-	300B	Clf+Dedup+PF	1024	A100	321h	1.32 Mil	M	Megatron
Yuan 1.0 [115]	arXiv'21	Apache-2.0	-	General	245B	✓	26k*	180B	Heur+Clf+Dedup	2128	GPU	-	-	D+T+P	-
Gopher [116]	arXiv'21	-	Google	General	280B	×	-	300B	QF+Dedup	4096	TPU v3	920h	13.19 Mil	D+M	JAX+Haiku
ERNIE 3.0 Titan [35]	arXiv'21	-	Baidu	General	260B	×	-	300B	Heur+Dedup	-	Ascend 910	-	-	D+M+P+D*	PaddlePaddle
GPT-NeoX-20B [118]	BigScience'22	Apache-2.0	EleutherAI	General	20B	✓	150k	825GB	None	96	40G A100	-	-	M	Megatron+DS+PyTorch
OPT [14]	arXiv'22	MIT	Meta	General	175B	✓	150k	180B	Dedup	992	80G A100	-	-	D+T	Megatron
BLOOM [13]	arXiv'22	RAIL-1.0	BigScience	General	176B	✓	-	366B	Dedup+PR	384	80G A100	2520h	3.87 Mil	D+T+P	Megatron+DS
Galactica [138]	arXiv'22	Apache-2.0	Meta	Science	120B	×	225k	106B	Dedup	128	80GB A100	-	-	-	Metaseq
GLaM [91]	ICML'22	-	Google	General	1.2T	×	600k*	600B	Clf	1024	TPU v4	-	-	M	GSPMD
LaMDA [140]	arXiv'22	-	Google	Dialog	137B	×	3M	2.81T	Filtered	1024	TPU v3	1384h	4.96 Mil	D+M	Lingvo
MT-NLG [117]	arXiv'22	Apache-v2.0	MS.+Nvidia	General	530B	×	-	270B	-	4480	80G A100	-	-	D+T+P	Megatron+DS
AlphaCode [132]	Science'22	Apache-v2.0	Google	Coding	41B	✓	205k	967B	Heur+Dedup	-	TPU v4	-	-	M	JAX+Haiku
Chinchilla [96]	arXiv'22	-	Google	General	70B	×	-	1.4T	QF+Dedup	-	TPUv4	-	-	-	JAX+Haiku
PaLM [15]	arXiv'22	-	Google	General	540B	×	255k	780B	Heur	6144	TPU v4	-	-	D+M	JAX+T5X
AlexaTM [122]	arXiv'22	Apache v2.0	Amazon	General	20B	×	500k	1.1T	Filtered	128	A100	2880h	1.47 Mil	M	DS
U-PaLM [124]	arXiv'22	-	Google	General	540B	×	20k	-	-	512	TPU v4	120h	0.25 Mil	-	-
UL2 [125]	ICLR'23	Apache-2.0	Google	General	20B	✓	2M	1T	-	512	TPU v4	-	-	M	JAX+T5X
GLM [33]	ICLR'23	Apache-2.0	Multiple	General	130B	×	-	400B	-	768	40G A100	1440h	3.37 Mil	M	-
CodeGen [130]	ICLR'23	Apache-2.0	Salesforce	Coding	16B	✓	650k	577B	Heur+Dedup	-	TPU v4	-	-	D+M	JAXFormer
LLaMA [127]	arXiv'23	-	Meta	General	65B	×	350k	1.4T	Clf+Heur+Dedup	2048	80G A100	504h	4.12 Mil	D+M	xFormers
PanGuΣ [92]	arXiv'23	-	Huawei	General	1.085T	×	-	329B	-	512	Ascend 910	2400h	-	D+OP+P+O+R	MindSpore
BloombergGPT [141]	arXiv23	-	Bloomberg	Finance	50B	×	139k	569B	Dedup	512	40G A100	1272h	1.97 Mil	M	PyTorch
Xuan Yuan 2.0 [142]	arXiv23	RAIL-1.0	Du Xiaoman	Finance	176B	✓	-	366B	Filtered	80GB	A100	-	-	P	DS
CodeT5+ [34]	arXiv'23	BSD-3	Salesforce	Coding	16B	✓	110k	51.5B	Dedup	16	40G A100	-	-	-	DS
StarCoder [137]	arXiv'23	OpenRAIL-M	BigCode	Coding	15.5B	✓	250k	1T	Dedup+QF+PF	512	80G A100	624h	1.28 Mil	D+T+P	Megatron-LM
LLaMA-2 [21]	arXiv'23	LLaMA-2.0	Meta	General	70B	✓	500k	2T	Minimal Filtering	-	80G A100	1.7Mh	-	-	-
PaLM-2 [123]	arXiv'23	-	Google	General	-	×	-	-	Ddedup+PF+QF	-	-	-	-	-	-

Fonte: Naveed et al. (2024).

Ademais, Ramos (2023) menciona que existe um imenso conjunto de novas ferramentas que estão surgindo com grande foco no uso de LLMs. Com base nisso, o autor propõe a Figura 13 com os principais LLMs aplicados à pesquisa acadêmica, que podem ser utilizadas também em outras áreas, com sua divisão por tarefa realizada.

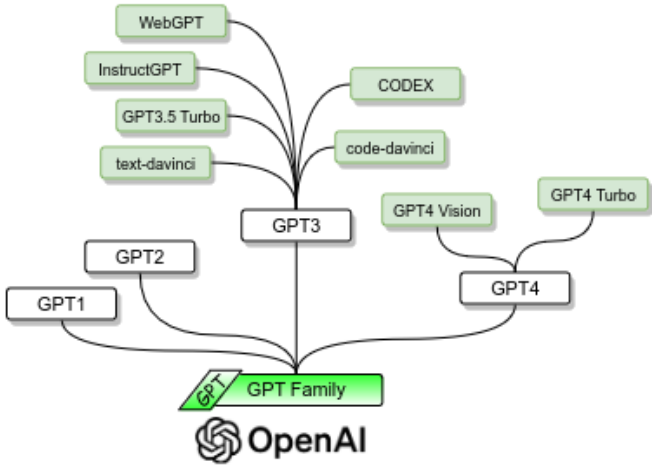
Figura 13 — Ferramentas de IA generativas aplicadas a pesquisa

Ferramentas / Tarefas	ChatGPT, Bing Chat	Klavier, ChatPDF, OpenRead	OKmaps, Litmaps, Connect Papers	Research Rabbit	Elicit	Consensus	Resoomer, Scholarcy
Busca assistida por IA	x	x	x	x	x	x	
Pergunta de pesquisa	x				x	x	
Brainstorm de pergunta	x				x	x	
Análise de redes de citação e cocitação			x	x			
Design de pesquisa	x				x		
Palavras-chave expandidas			x		x		x
Perguntas e respostas sobre artigo	x	x			x	x	
Geração de texto de resumo	x	OpenRead			x	x	x
Exportação			x	x	x		x
Acesso ao PDF	No Bing		x		x	x	
Consenso da literatura						x	

Fonte: Ramos (2023).

Entre os principais LLMs, destaca-se o GPT (Transformador Generativo Pré-Treinado), que consiste em um modelo ou, mais precisamente, uma família de modelos de linguagem baseados na arquitetura *Transformer* e estruturados como decodificadores, desenvolvidos pela OpenAI. Essa família inclui versões como GPT-1, GPT-2, GPT-3, InstructGPT, ChatGPT, GPT-4, Codex e WebGPT, conforme ilustrado na Figura 14.

Figura 14 — Família GPT



Fonte: Minaee et al. (2024).

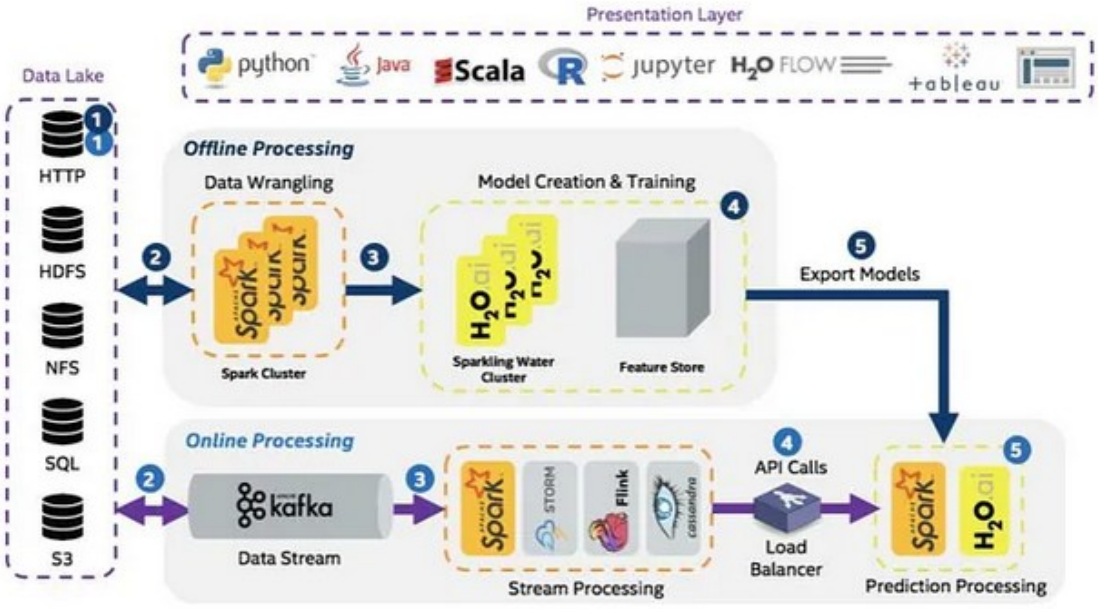
Minaee et al. (2024) informam que os primeiros modelos, como GPT-1 e GPT-2, são de código aberto, enquanto os demais estão disponíveis apenas por meio de API. Ainda segundo os autores, o GPT-3 é um modelo de linguagem autorregressivo pré-treinado com 175 bilhões de parâmetros, sendo considerado o primeiro LLM por demonstrar, pela primeira vez, habilidades não observadas em modelos de linguagem pré-treinados (PLMs) menores. Além disso, apresenta a capacidade emergente de aprendizagem em contexto, permitindo sua aplicação em diferentes contextos.

O Codex foi lançado pela OpenAI em março de 2023, sendo um modelo de programação de uso geral capaz de interpretar linguagem natural e gerar código em resposta. O WebGPT é outro descendente do GPT-3, ajustado para responder a perguntas abertas com o auxílio de um navegador baseado em texto, facilitando a busca e navegação na web. Já o InstructGPT foi desenvolvido com o objetivo de alinhar os modelos de linguagem às intenções dos usuários em uma ampla gama de tarefas, sendo ajustado com base em feedback humano. Por fim, o GPT-3 foi refinado com base em conjuntos de dados compostos por respostas classificadas por humanos, utilizando técnicas de aprendizagem por reforço (MINAEE et al., 2024).

Conforme destacado por Minaee et al. (2024), o grande marco no que tange o desenvolvimento dos LLMs foi o lançamento do ChatGPT (*Chat Generative Pre-trained Transformer)*, em 30 de novembro de 2022. Trata-se de um chatbot que permite aos usuários conduzir conversas para a realização de diversas tarefas, como responder perguntas, buscar informações e produzir resumos de texto, entre outras funcionalidades. O sistema foi inicialmente baseado no GPT-3.5 e, posteriormente, no GPT-4, com evoluções contínuas dentro dessa família de modelos, sendo o mais recente e poderoso, o GPT-5.4.

Quanto à arquitetura do h2o, ela é aberta, desenvolvida em Java, com interoperabilidade com Python e R, e conta com mais de 15 algoritmos implementados, tanto supervisionados quanto não supervisionados. A Figura 15 demonstra as tecnologias envolvidas, incluindo data lakes para armazenamento de dados, processamento, modelos de treinamento e as linguagens de programação que podem ser utilizadas.

Figura 15 — Tecnologias envolvidas no h2o



Fonte: H2O.ai (2024).

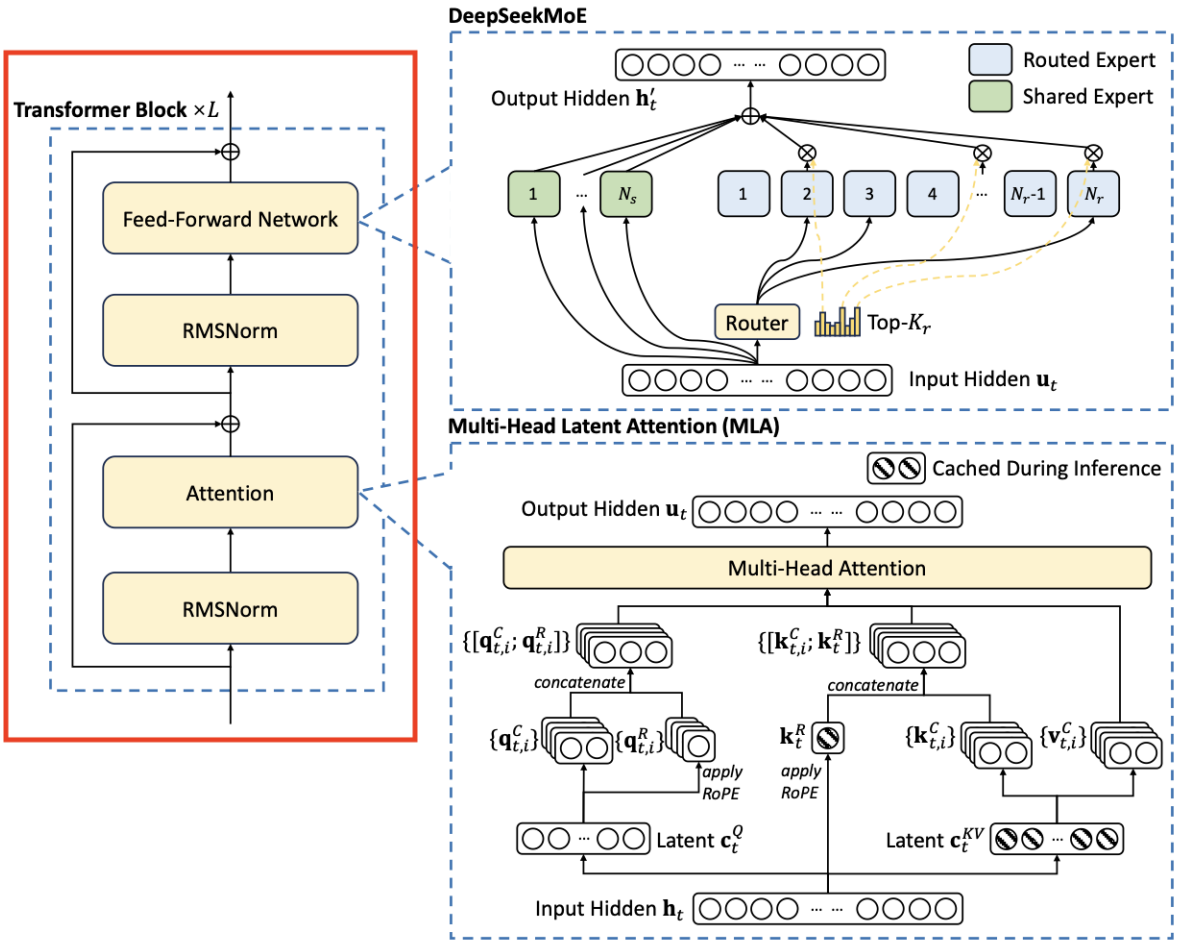
Conforme portal h2o (2024), a plataforma é utilizada por mais de 14.000 organizações e centenas de milhares cientistas de dados, em vista de sua robustez. Quanto ao ChatGPT, ele conseguiu atingir a marca de 1 milhão de usuários em apenas 5 dias, sendo o produto de maior crescimento mais rápido da história da internet até então, segundo registros amplamente documentados pela imprensa especializada (REUTERS, 2023).

Além disso, o LLaMa da Meta AI, foi introduzido como uma alternativa *open-source*, alcançando grande destaque pela sua arquitetura eficiente e capacidade de competir com modelos robustos do mercado. Sua última versão, LLaMa 3.2, foi lançada em setembro de 2024 e possui 4 versões que variam conforme a quantidade de parâmetros pretendidos. São as mais robustas, com 11 bilhões e 90 bilhões, e outras duas com 1 bilhão e 3 bilhões, focadas para operar em dispositivos móveis (Knight, 2024).

Outro modelo recente que vem ganhando repercussão é o DeepSeek-V2, desenvolvido pela empresa chinesa DeepSeek. Lançado inicialmente com foco em pesquisa e desenvolvimento, o modelo destaca-se por apresentar uma arquitetura robusta com capacidade de raciocínio avançado e forte desempenho em *benchmark*s ocidentais. Segundo reportagem do *The Guardian* (2025), o DeepSeek representa uma aposta estratégica da China para competir com os modelos mais avançados do Ocidente, especialmente em aplicações de linguagem natural, codificação e tradução.

Além disso, o modelo utiliza uma abordagem de treinamento com *Mixture-of-Experts* (MoE) e foi testado com 236 bilhões de parâmetros ativos, o que lhe permite balancear eficiência computacional e qualidade de geração textual. Sua abertura ao público e foco em pesquisa reforçam seu papel como um dos principais representantes da nova geração de LLMs de código aberto. A Figura 16 demonstra o diagrama dessa arquitetura, destacando os componentes principais como o *framework Mixture of Experts* (MoE) e o mecanismo de *Multi-Head Latent Attention* (MLA) (ZHANG et al., 2025).

Figura 16 — Diagrama de arquitetura dos modelos DeepSeek V3 e R1.



Fonte: FIREWORKS AI (2025).

2.4 OPEN SOURCE

De acordo com a Open Source Initiative (2024), o termo *open source* significa código aberto, indicando que o código fonte de determinado software é público e pode ser modificado pela comunidade, promovendo inovações e melhorias. Dessa forma, não há custo de licença, permitindo sua utilização por empresas e indivíduos interessados. Entre as tecnologias de código aberto mais conhecidas estão o Linux, o LibreOffice e o editor de imagens GIMP.

Essa nomenclatura foi criada em 1998, durante uma reunião que contou com a participação de personalidades que se tornariam referência no tema, como Todd Anderson, Chris Peterson, Larry Augustin, Jon “Maddog”, Sam Ockman e Eric Raymond (OPEN SOURCE INITIATIVE, 2024).

Nesse contexto, o compartilhamento de código possibilita que diversas pessoas proponham inovações às tecnologias, o que é especialmente relevante no caso de áreas ainda

em desenvolvimento, como a inteligência artificial generativa. Isso contribui para maior transparência em seu funcionamento e nas bases de dados utilizadas em seu treinamento. Por outro lado, o fato de o código ser aberto e público pode facilitar que indivíduos mal-intencionados identifiquem bugs e explorem falhas no sistema, visando ganhos financeiros ou reconhecimento.

Além disso, conforme apontam Wang et al. (2025), o movimento *open source* tem desempenhado um papel fundamental no avanço dos LLMs, ao promover maior acessibilidade, auditabilidade e reprodutibilidade científica. A disponibilização pública dos modelos e dos dados de treinamento permite uma avaliação mais transparente de seu desempenho e de suas limitações, além de acelerar a inovação colaborativa no campo da inteligência artificial, possibilitando que pesquisadores ao redor do mundo contribuam com melhorias significativas.

É pertinente estabelecer uma distinção técnica no ecossistema de modelos abertos. Embora o termo 'O*pen Source*' seja amplamente utilizado na literatura, modelos recentes, como a família LLaMA e o GPT-OSS, operam sob o paradigma de *Open Weights* (Pesos Abertos). Nesse formato, os desenvolvedores disponibilizam os parâmetros treinados da rede neural para inferência e ajustes finos (*fine-tuning*) locais, permitindo a execução do modelo em infraestrutura própria. No entanto, diferentemente de projetos estritamente alinhados à definição da OSI, os conjuntos de dados de treinamento (*datasets*) e o código completo de pré-processamento frequentemente permanecem proprietários ou sob licenças restritivas.

Cabe registrar, contudo, que a literatura especializada frequentemente utiliza o termo open source de maneira ampla, abrangendo também o paradigma de open weights (pesos abertos), no qual os parâmetros treinados de um modelo são disponibilizados publicamente sem que o código de treinamento ou os dados sejam necessariamente abertos (WANG et al., 2025). Neste trabalho, adota-se essa acepção mais abrangente, alinhada ao uso corrente da comunidade de pesquisa em LLMs, uma vez que os modelos avaliados, como LLaMA e GPT-OSS, operam sob licenças de pesos abertos.

2.4.1 Casos de Sucesso e Limitações Open Source

A Fundação Linux publicou em dezembro de 2023 uma extensa pesquisa sobre o comportamento e expectativas a respeito da IA generativa, na qual cerca de 50% das pessoas que responderam desejam usar IA em sua empresa, 63% acham a tecnologia extremamente importante para o futuro, entre outros. Já em relação a IA generativa de código aberto, 41% das organizações afirmaram preferir uma tecnologia de código aberto, contra 9% que preferem

tecnologias proprietárias. De acordo com os entrevistados, a tecnologia aberta também possui maior inovação, colaboração e facilidade de integração, além de facilitar a transparência e a aquisição dos dados.

Logo, percebe-se que as empresas enxergam grande potencial nas tecnologias de código aberto, embora, na prática, elas ainda não sejam tão utilizadas quanto as proprietárias (por exemplo, Linux versus Windows). No entanto, devido à colaboração inerente a esse modelo, espera-se maior inovação e transparência na área.

Em relação à importância do código aberto nos modelos de linguagem, Bussler (2024) argumenta que ele promove transparência, inovação colaborativa e controle sobre o desenvolvimento, permitindo que pesquisadores e desenvolvedores adaptem e aprimorem os modelos conforme suas necessidades. Além disso, o código aberto amplia a acessibilidade para organizações menores, possibilitando a criação de soluções customizadas, ao contrário dos modelos proprietários, que podem apresentar limitações quanto à privacidade, ao controle e aos custos.

Silvestre (2024) reforça que o *open source* é crucial para a inteligência artificial, pois viabiliza a criação de modelos menores e especializados, adaptados a necessidades específicas das empresas. Essa abordagem democratiza o acesso e potencializa a especialização, permitindo que as organizações utilizem seus próprios dados para inovar.

Por outro lado, embora os modelos de código aberto ofereçam benefícios significativos, é necessário considerar que nem todos os modelos anunciados como *open source* são, de fato, totalmente abertos. Muitas iniciativas em IA generativa ainda carecem de um consenso claro sobre o que caracteriza o *open source*. Em alguns casos, empresas utilizam o termo como estratégia de marketing, sem compartilhar elementos essenciais do código ou conceder plena liberdade de adaptação e redistribuição. Essa falta de transparência e padronização pode limitar os benefícios esperados, perpetuando a exclusividade e restringindo o progresso colaborativo associado ao código aberto (MIT TECHNOLOGY REVIEW, 2024).

Dessa forma, observa-se que, embora os modelos de código aberto ofereçam vantagens como flexibilidade e redução de custos, também apresentam desafios, como a necessidade de recursos e habilidades técnicas avançadas para sua implementação e manutenção, além de possíveis questões relacionadas à segurança e ao suporte limitado. Em contrapartida, LLMs proprietários, apesar do custo e da menor flexibilidade, geralmente oferecem suporte técnico robusto, garantias de segurança e atualizações regulares, o que pode ser essencial para empresas

que necessitam de soluções rápidas e confiáveis, sem a capacidade de gerenciar projetos de código aberto (CLOUDSMITHS, 2024).

2.5 RELAÇÃO REVISÃO TEÓRICA COM FRAMEWORK PROPOSTO

A revisão bibliográfica apresentou os conceitos centrais sobre grandes modelos de linguagem (LLMs), o movimento *open-source*, diferenças entre soluções proprietárias e abertas, além de casos de sucesso e limitações práticas desses modelos. A partir dessa análise, identificaram-se lacunas operacionais relevantes para estudos comparativos: a necessidade de um *pipeline* padronizado para execução via APIs, mecanismos explícitos para preservação e versionamento dos artefatos brutos, procedimentos automáticos para tratamento de falhas e práticas de governança e rastreabilidade dos dados de execução.

Com base nessas lacunas, o presente trabalho propõe um *framework* experimental que operacionaliza essas práticas, integrando execução por API, *logging* estruturado, preservação dos resultados brutos e infraestrutura de reprodutibilidade. A Metodologia (Capítulo 3) descreve os procedimentos, parâmetros e componentes do *pipeline* projetados para garantir a consistência e a rastreabilidade necessárias à replicação dos experimentos.

3 METODOLOGIA

Este capítulo descreve a metodologia adotada para o desenvolvimento do framework de comparação de LLMs, abrangendo as métricas, os benchmarks, as ferramentas de software e integrações empregadas, os modelos de linguagem *open source* e proprietários selecionados, bem como os procedimentos adotados para execução e análise comparativa.

Inicialmente, foi realizada uma revisão bibliográfica, na qual, por meio da leitura de livros e artigos, construiu-se o referencial teórico necessário para possibilitar uma análise crítica do estado da arte das inteligências artificiais generativas de texto. Para isso, foram buscados artigos em bases de dados como IEEE Explorer e Google Scholar. Por se tratar de uma tecnologia recente, não houve delimitação quanto ao período de abrangência da pesquisa, sendo o principal critério de seleção a relevância do trabalho e sua correlação com o tema investigado.

Além disso, a pesquisa teve caráter exploratório, investigando tendências, desafios e avanços em inteligências artificiais generativas de texto, com foco em modelos de linguagem de código aberto e proprietários. Esse estudo proporcionou uma visão abrangente do campo e orientou a seleção dos modelos analisados e comparados.

Com o referencial teórico estabelecido, este capítulo apresenta o processo metodológico utilizado na construção e execução do framework experimental de comparação, detalhando as métricas e os benchmarks (Seção 3.1), as ferramentas e integrações (Seção 3.2), os modelos selecionados (Seção 3.3), o conjunto de *prompts* e os procedimentos de execução (Seção 3.4), além dos aspectos de reprodutibilidade, versionamento de resultados e governança de dados (Seção 3.5).

Por fim, os resultados obtidos a partir da execução do framework e sua aplicação prática na comparação entre diferentes Large Language Models (LLMs) são apresentados no Capítulo 4, permitindo avaliar, de forma objetiva e padronizada, o comportamento dos modelos analisados.

3.1 MÉTRICAS E BENCHMARKS

A fim de garantir uma análise objetiva e reproduzível entre as respostas dos diferentes modelos de linguagem, adotam-se métricas e *benchmark*s consolidados na literatura para avaliação automática de textos gerados. As ferramentas selecionadas abrangem aspectos lexicais e semânticos das saídas, bem como *benchmark*s de conhecimento e raciocínio.

3.1.1 ROUGE-1/ROUGE-2/ROUGE-L

A métrica *Recall-Oriented Understudy for Gisting Evaluation* (ROUGE) é uma das mais utilizadas na avaliação automática de tarefas de geração de texto, especialmente em sistemas de sumarização automática. Essa métrica mensura a sobreposição de unidades linguísticas (como n-gramas, palavras e sequências) entre um texto gerado automaticamente e uma referência produzida por humanos (LIN, 2004).

Neste trabalho, empregam-se múltiplas variantes (1/2/L), pois isso permite avaliar uma maior subsequência comum entre os textos, contribuindo para a mensuração da fluidez e da estrutura lógica da resposta gerada. Além disso, essa abordagem mostrou-se essencial para captar semelhanças superficiais entre os textos produzidos pelos modelos e as respostas de referência.

3.1.2 BLEU

A métrica *Bilingual Evaluation Understudy* (BLEU), proposta por Papineni et al. (2002), é uma das métricas mais amplamente utilizadas para avaliação automática de textos gerados por sistemas de linguagem natural, originalmente concebida para tarefas de tradução automática. Seu cálculo baseia-se na precisão de *n*-gramas (sequências contíguas de *n* palavras) presentes tanto na resposta gerada quanto no texto de referência, aplicando uma penalização por brevidade (*brevity penalty*) para evitar que respostas excessivamente curtas sejam artificialmente favorecidas.

Neste trabalho, o uso do BLEU complementa a análise do ROUGE ao focar na fidelidade estrutural e na assertividade dos termos utilizados pelos modelos. Essa abordagem permite verificar o quanto as sequências de palavras geradas coincidem com os padrões esperados nos gabaritos, sendo essencial para validar a precisão técnica das saídas em tarefas de resposta curta e direta.

3.1.3 BERTScore

O BERTScore é uma métrica de avaliação baseada no sentido (semântica) dos textos, que utiliza representações vetoriais provenientes de modelos de linguagem pré-treinados. Diferente de métricas que buscam apenas palavras idênticas, ele calcula a similaridade entre os vetores dos componentes do texto (*tokens*), levando em conta o contexto em que estão inseridos (ZHANG et al., 2020).

No presente trabalho, adotou-se o modelo RoBERTa-large (LIU et al., 2019) como estrutura base para a geração desses vetores. Essa abordagem possibilita avaliar a semelhança de significado entre a resposta gerada e a referência, mesmo diante de variações no vocabulário. No contexto deste projeto, o BERTScore mostrou-se fundamental para identificar respostas que, embora apresentem palavras diferentes, possuem significados equivalentes, garantindo uma análise mais sensível ao contexto técnico.

3.1.4 HellaSwag

O HellaSwag é um *benchmark* criado por Zellers et al. (2019), com o objetivo de avaliar o raciocínio de senso comum dos modelos de linguagem. Ele se baseia em uma tarefa de escolher, entre quatro alternativas, qual a continuação mais plausível para um enunciado.A escolha do HellaSwag se deve ao fato que ele não mede apenas similaridade lexical, mas também habilidade de inferência contextual e causal. Isto significa que, é possível distinguir modelos que apenas reproduzem padrões linguísticos daqueles que demonstram maior compreensão semântica.

3.1.5 MMLU

O *Massive Multitask Language Understanding* (MMLU) consiste em um benchmark de conhecimento multitarefa, composto por 57 disciplinas, voltado à avaliação de modelos de linguagem quanto à sua capacidade de compreender e aplicar conhecimentos gerais em diferentes áreas (HENDRYCKS et al., 2021).

Neste trabalho, optou-se pela utilização do MMLU justamente por sua capacidade de oferecer uma avaliação abrangente, não restrita a tarefas específicas ou a domínios isolados. Esse benchmark possibilita verificar se o modelo, de fato, incorpora conhecimento diversificado e habilidades de raciocínio, além de facilitar a comparação com outros estudos que também adotam essa mesma referência.

3.1.6 EvidentlyAI

O EvidentlyAI é uma ferramenta *open source* para monitoramento e análise de modelos de *machine learning*, permitindo avaliação quantitativa da performance e qualidade dos modelos em produção ou em testes comparativos. No presente estudo, foi utilizada para mensurar métricas objetivas que auxiliaram na comparação entre os modelos de linguagem, fornecendo uma base robusta para análise de resultados (EVIDENTLYAI, 2024).

A escolha do EvidentlyAI se deve à sua capacidade de gerar relatórios visuais e estatísticos detalhados sobre as diferenças entre as respostas dos modelos, como variação de comprimento, distribuição de tokens e diversidade lexical. Essa ferramenta complementa as métricas formais ao oferecer uma visão exploratória do comportamento dos modelos.

3.1.7 Normalização e Interpretação dos Scores

As métricas apresentadas neste trabalho (BLEU, ROUGE, BERTScore) foram submetidas a uma normalização Min–Max com o objetivo de permitir comparação intra-execução entre modelos com escalas e distribuições distintas. A normalização é aplicada apenas para fins comparativos internos do *framework*. Os valores brutos são preservados e versionados no repositório do projeto para reprodução.

É importante observar que métricas baseadas em sobreposição lexical, como BLEU e ROUGE, tendem a produzir valores baixos quando aplicadas a *prompt*s abertos e respostas livres, pois penalizam variações lexicais legítimas, mesmo quando há equivalência semântica. Por esse motivo, as interpretações no presente trabalho priorizam a variação relativa entre modelos, o BERTScore (avaliação semântica) e os relatórios do EvidentlyAI, evitando conclusões absolutas baseadas em um único indicador.

3.2 FERRAMENTAS E INTEGRAÇÕES

Esta seção apresenta as bibliotecas, os *framework*s e os serviços utilizados na preparação de dados, execução dos modelos via API, cálculo de métricas e visualização de resultados, o que padroniza o *pipeline* experimental e favorece a reprodutibilidade do estudo.

3.2.1 Processamento de Dados

- **Pandas**: manipulação e análise de dados estruturados; organização de execuções, métricas e consolidação de resultados para análise.
- **Numpy**: suporte a computação numérica e operações vetorizadas para transformações auxiliares nas etapas de avaliação e análise descritiva.
- **Scikit-learn**: utilizado pontualmente para métricas estatísticas, normalização e análises simples que apoiam a comparação entre modelos.

3.2.2 APIs e Integração

- **Groq API**: execução de modelos *open source* com baixa latência; utilizada para padronizar chamadas e parâmetros sem depender de infraestrutura local.
- **Google Generative AI API**: acesso aos modelos proprietários Gemini, mantendo uniformidade de *prompt*s, limites e coleta de respostas para comparação direta.
- **Requests**: cliente HTTP para chamadas diretas às APIs, com controle de erros e *timeouts*.
- **Python-dotenv**: Gerenciamento de variáveis de ambiente e chaves de API, isolando credenciais e facilitando a configuração do ambiente experimental.

3.2.3 Avaliação e Métricas

- ***Evaluate* (Hugging Face)**: responsável pelo cálculo padronizado das métricas ROUGE e BLEU, garantindo compatibilidade com trabalhos correlatos.
- ***Transformers* e *sentence-transformers***: utilizadas para tokenização e geração de *embeddings* semânticos, fundamentais para o cálculo do BERTScore.
- **EvidentlyAI**: adotado como ferramenta complementar de análise estatística, fornecendo relatórios visuais sobre variação de respostas, diversidade lexical e detecção de *drift*.

3.2.4 Visualização e Análise

- **Matplotlib**: gráficos de distribuição e comparação de resultados consolidados após a etapa de avaliação definida em 3.1, integrados aos relatórios do projeto.
- **Seaborn**: visualizações estatísticas para inspeção de tendências, correlações e comparações entre modelos, apoiando a análise descritiva dos resultados.

3.3 MODELOS DE LINGUAGEM UTILIZADOS

3.3.1 Modelos Open Source (via Groq API)

Os modelos LLaMA 3.1 8B Instant e LLaMA 3.3 70B Versatile, desenvolvidos pela Meta, integram a família LLaMA (Large Language Model Meta AI), reconhecida como uma das mais avançadas e utilizadas no campo dos modelos de linguagem de código aberto. O LLaMA 3.1 8B Instant foi projetado para aplicações que demandam maior eficiência e baixa latência, enquanto o LLaMA 3.3 70B Versatile apresenta maior capacidade de raciocínio e

qualidade de geração, sendo mais adequado para tarefas complexas que exigem maior contexto. Ambos compartilham a mesma arquitetura baseada em Transformer, diferenciando-se principalmente pela escala de parâmetros (8 bilhões versus 70 bilhões) e, consequentemente, pelo desempenho em tarefas de geração (META, 2024; KNIGHT, 2024).

Os modelos GPT-OSS 20B e GPT-OSS 120B, por sua vez, são modelos de pesos abertos (open-weight) inspirados na família GPT, lançados pela OpenAI em 2025. O GPT-OSS 20B oferece um equilíbrio entre custo computacional e desempenho, sendo indicado para aplicações mais leves, enquanto o GPT-OSS 120B representa uma versão de maior escala, voltada a cenários que exigem elevada capacidade de geração textual e raciocínio mais avançado. A principal distinção entre ambos reside no número de parâmetros, fator que impacta diretamente tanto a qualidade das respostas quanto o custo de execução (OPENAI, 2025).

Por fim, o Qwen 3 32B, desenvolvido pela Alibaba, faz parte da família Qwen 3 lançada em 2025 e foi projetado para uso multilíngue, destacando-se pelo bom desempenho em português. Sua arquitetura, também baseada em Transformer, incorpora variantes densas e do tipo Mixture of Experts, otimizadas para suportar tarefas de geração e compreensão em diferentes idiomas, além de operar com janelas de contexto extensas. Tais características tornam o Qwen 3 32B especialmente relevante em cenários de análise comparativa multicultural (YANG et al., 2025).

3.3.2 Modelos Proprietários (via Google AI API)

O Gemini 2.5 Flash Lite e o Gemini 3.0 Preview são uma variante otimizada da família Gemini, voltada para aplicações que exigem velocidade e baixa latência. Sua arquitetura proprietária permite uma janela de contexto de até 1 milhão de tokens, com suporte a *thinking mode*, multimodalidade (texto, imagem, áudio, vídeo) e integração de ferramentas como execução de código e buscas externas. O modelo se destaca também por ser de baixo custo relativo (mais eficiente em termos de custo por token) comparado a outras variantes anteriores. (GOOGLE, 2026).

3.4 PROCEDIMENTOS E PROMPTS UTILIZADOS

Esta seção descreve como os *prompt*s foram estruturados, aplicados e avaliados no *pipeline* experimental, detalhando o conteúdo, o fluxo de execução definido no projeto e os critérios de análise utilizados neste estudo.

3.4.1 Visão Geral

Esta pesquisa segue um *pipeline* composto por cinco etapas principais: (1) inicialização de configurações e chaves de API; (2) execução sequencial de *prompt*s em diferentes modelos; (3) avaliação com métricas e *benchmark*s; (4) consolidação de resultados e relatórios; (5) análise comparativa entre execuções. Os dados de entrada incluem os *prompt*s estruturados e os *benchmark*s padronizados disponíveis no repositório, garantindo a reprodutibilidade do experimento. A partir desta visão geral, serão detalhados os procedimentos de execução, o design dos *prompt*s e os *benchmark*s utilizados para mensuração objetiva.

3.4.2 Objetivo e escopo dos prompts

A estrutura dos *prompt*s utilizados no sistema é composta por identificadores únicos, tipo de conteúdo, enunciado, categoria temática e idioma. Essa organização padronizada permite a ingestão automatizada por *pipeline*s de processamento, além de facilitar processos de auditoria e rastreabilidade. O *framework* aceita *prompts* multilíngues e preserva o idioma da entrada para a análise comparativa. A classificação por categoria e idioma contribui significativamente para a análise comparativa entre modelos, especialmente em subdomínios semânticos distintos, promovendo maior precisão na avaliação.

As respostas de referência (*reference*) utilizadas no cálculo das métricas automáticas foram geradas por um modelo de linguagem de maior capacidade, o GPT-5 (OpenAI), com o objetivo de fornecer uma base semântica de alta qualidade para comparação. Essa abordagem é recorrente em estudos de avaliação automatizada de LLMs, uma vez que permite a obtenção de referências consistentes sem depender de anotação humana extensiva, reduzindo o viés subjetivo na construção do gabarito. É reconhecido, contudo, que o uso de respostas geradas por modelo implica limitações inerentes, discutidas na Seção 5.

Para padronizar a entrada de dados no *framework*, utilizou-se um arquivo no formato JSON contendo os dados essenciais para as requisições. A seguir, o Quadro 1 ilustra uma amostra da estrutura dos dados submetidos aos modelos.

Quadro 1 — Trecho de entrada JSON dos prompts

```
{      "id": 1,
      "type": "explanatory",
      "prompt": "O que é inteligência artificial e como ela está transformando o
mundo? Forneça uma explicação detalhada com exemplos práticos.",
      "category": "AI_Fundamentals",
      "language": "pt"
```

```
    },
{     "id": 2,
      "type": "explanatory",
      "prompt": "Explique a diferença entre machine learning e deep learning,
incluindo casos de uso específicos para cada abordagem.",
      "category": "AI_Fundamentals",
      "language": "pt"
    },
{     "id”: 3,
      “type”: “explanatory”,
      “prompt”: “Como funcionam os modelos de linguagem como GPT e LLaMA? Descreva
a arquitetura e o processo de treinamento.”,
      “category”: “AI_Fundamentals”,
      “language”: “pt”
    },
```

Fonte: Elaborado pelo autor (2026).

Nota: Trecho reduzido para fins ilustrativos.

3.4.3 Benchmarks acadêmicos

Já para os *benchmark*s, o conjunto de dados utilizados reúne tarefas de desafios MMLU (*Massive Multitask Language Understanding*) e HellaSwag com questões, alternativas e gabaritos, o que possibilita a mensuração objetiva de conhecimento e capacidade de raciocínio dos modelos utilizados.

Em MMLU, há itens por disciplina com identificadores e respostas corretas, enquanto em HellaSwag oferece contextos narrativos com alternativas plausíveis para seleção baseada em coerência. Em seguida, o Quadro 2 demonstra um trecho da estrutura dos dados submetidos aos modelos com desafios MMLU e HellaSwag.

Quadro 2 — Trecho de entrada JSON dos prompts de *benchmarks*

```
(...)
    "mmlu": {
      "name": "MMLU",
      "description": "Massive Multitask Language Understanding",
      "subjects": {
        "abstract_algebra": [
          {
            "id": "mmlu_001",
            "question": "Which of the following is a group?",
            "choices": [
              "A) The set of all integers under addition",
```

```
              "B) The set of all real numbers under multiplication",
              "C) The set of all positive integers under subtraction",
              "D) The set of all rational numbers under division"
            ],
            "answer": "A",
            "subject": "abstract_algebra"
          },
(...)
    "hellaswag": {
      "name": "HellaSwag",
      "description": "Commonsense Reasoning",
      "questions": [
        {
          "id": "hellaswag_001",
          "context": "A man is sitting in a chair reading a book. The book is very
interesting and he is completely absorbed in it.",
          "question": "What happens next?",
          "choices": [
            "A) He continues reading the book",
            "B) He gets up and walks away",
            "C) He falls asleep",
            "D) He throws the book away"
          ],
          "answer": "A"
        },
(...)
```


Fonte: Elaborado pelo autor (2026).

Nota: Trecho reduzido para fins ilustrativos.

3.4.4 Padrões de respostas e avaliação

O processo de execução inicia-se com a leitura dos dados de entrada, seguido pela orquestração das chamadas aos modelos, coleta das respostas geradas e cálculo das métricas de desempenho. Assim, cada etapa resulta em artefatos organizados por instância de execução, o que permite rastreabilidade detalhada e facilita comparações entre diferentes versões ou configurações.

A documentação técnica do projeto estabelece uma separação clara entre os módulos de execução, avaliação e análise, o que contribui para a redução da variabilidade operacional e assegura a reprodutibilidade dos experimentos em ambientes distintos. Essa modularidade também facilita o tratamento de exceções e a manutenção da integridade dos dados processados.

As respostas geradas pelos modelos são avaliadas com base em métricas que contemplam tanto a qualidade textual quanto a coerência semântica. Indicadores de similaridade e acurácia em questões de múltipla escolha são utilizados em conjunto, oferecendo uma visão complementar entre a capacidade redacional e o desempenho factual. Essa abordagem permite uma análise mais abrangente da competência dos modelos em tarefas de linguagem natural.

Os resultados e metadados de cada execução são armazenados em diretórios específicos, com registros detalhados para auditoria e arquivos consolidados para análise comparativa. Essa estrutura padronizada viabiliza a reexecução controlada dos experimentos, facilita a identificação de anomalias e sustenta a geração de relatórios reprodutíveis com alto grau de confiabilidade.

A integração das ferramentas é realizada em um fluxo único que conecta leitura dos insumos, execução dos modelos e cálculo das métricas, com separação de responsabilidades entre os módulos. O uso de controle de versão via GitHub garante rastreabilidade das alterações metodológicas e dos dados utilizados, além de permitir a integração contínua de melhorias no sistema, promovendo evolução incremental e sustentada da plataforma de avaliação.

3.5 REPRODUTIBILIDADE E GOVERNANÇA DE DADOS

A confiabilidade dos resultados obtidos em experimentos computacionais depende diretamente da capacidade de reproduzi-los em diferentes ambientes e momentos. Este projeto adota práticas consolidadas de reprodutibilidade e governança de dados, assegurando rastreabilidade, integridade e transparência em todas as etapas do *pipeline*.

A reprodutibilidade do projeto é garantida por uma estrutura modular e documentada, que organiza o fluxo de execução em etapas bem definidas, sendo elas: leitura de insumos; execução dos modelos via API; avaliação das respostas e análise dos resultados. Tal padronização, aliada à gestão estrita de dependências técnicas, minimiza discrepâncias entre execuções e garante a comparabilidade dos *benchmarks*.

O controle de versão via repositório remoto, permite rastrear alterações no código-fonte, nos arquivos de configurações e dados de entrada, com históricos detalhados. Isso resulta na viabilização da auditoria técnica e na facilitação da evolução contínua do sistema. Além disso, como recurso fundamental para a replicação deste estudo, o roteiro de configuração do ambiente, o código-fonte integral e as instruções de execução encontram-se documentados no Apêndice A. Por fim, através deste artefato, viabiliza-se a validação da consistência dos dados

e a identificação de variações, em conformidade com as boas práticas de engenharia de software e pesquisa científica.

4 DESENVOLVIMENTO

Este capítulo aborda o desenvolvimento prático do sistema de comparação de modelos de linguagem (LLMs), descrevendo a implementação do *pipeline* experimental, os procedimentos adotados nos testes e a aplicação das métricas definidas previamente na metodologia. Com base na estrutura e ferramentas apresentadas no Capítulo 3, foram realizados experimentos controlados com o propósito de avaliar o desempenho de diferentes modelos, incluindo *open source* e proprietários.

4.1 VISÃO GERAL

O projeto teve como foco desenvolver uma estrutura experimental para comparar de forma objetiva e reprodutível, o desempenho dos diferentes modelos de linguagem. Para isso, foi criado um *framework* modular, implementado em Python, que integra as principais ferramentas, APIs e bibliotecas descritas anteriormente.

A arquitetura do sistema foi planejada para garantir clareza, reprodutibilidade e facilidade de manutenção. Cada componente do *framework* cumpre uma função específica, desde a coleta e padronização das respostas geradas pelos modelos até o cálculo automático das métricas e geração de relatórios de análise comparativa.

Além disso, visando garantir a reprodutibilidade técnica do projeto, o código-fonte foi disponibilizado em um repositório remoto público, cuja documentação e link de acesso estão disponíveis no Apêndice A. Dessa forma, possibilita-se que outros pesquisadores repliquem e validem os experimentos de maneira autônoma.

4.2 ARQUITETURA DO SISTEMA

A arquitetura do sistema foi projetada para ser modular, extensível e transparente, permitindo que novos modelos, métricas ou *benchmark*s sejam facilmente adicionados ao *pipeline*. O fluxo operacional é coordenado pelo arquivo principal main.py, que orquestra a execução do *pipeline*, e pelo módulo de análise analysis/analysis.py, responsável pela coordenação das etapas de avaliação e consolidação dos resultados. As principais responsabilidades dos componentes são:

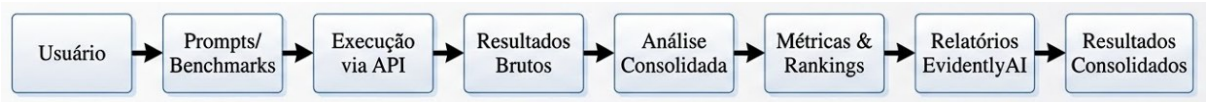
- **Configurações e chaves de API:** carregadas a partir de config.py e variáveis de ambiente (.env);
- **Execução dos modelos:** realizada via Groq API (para modelos *open source*) e Google Generative AI API (para os modelos Gemini). Além disso, foi incluído controle de

pacing, com *delay* entre *prompts* e atraso adicional para Gemini, medida que reduz falhas de cota e reforça a estabilidade do *pipeline* sem alterar a lógica experimental;

- **Resiliência de execução e tratamento de falhas:** o módulo models.py implementa mecanismos para lidar com erros de API, como *rate limit*, *timeout* e indisponibilidade momentânea. A estratégia adotada faz com que o sistema espere intervalos cada vez maiores entre novas tentativas, o que ajuda a evitar sobrecarga de requisições.
- **Armazenamento dos resultados brutos:** em formato CSV e JSON, organizados em diretórios específicos dentro de /results, com metadados de execução (*timestamp*, modelo, *prompt*, status, tempo de resposta);
- **Análise e orquestração:** o arquivo analysis/analysis.py centraliza a análise consolidada — chama os módulos de métrica (bleu_rouge.py, bertscore.py, mmlu.py, hellaswag.py), aplica normalizações, agrega métricas por execução e produz artefatos intermediários (JSONs e tabelas) para posterior inspeção;
- **Geração de relatórios e visualização:** a partir dos artefatos consolidados, analysis/evidently_reports.py gera relatórios HTML com EvidentlyAI (qualidade de dados, *drift*, distribuições) e analysis/ranking_system.py organiza as métricas processadas em estruturas comparativas, permitindo visualização unificada dos resultados.

A Figura 17 apresenta o fluxo geral do funcionamento do *framework*, destacando as principais etapas do processo de execução e análise.

Figura 17 — Fluxo geral do *framework*



Fonte: Elaborado pelo autor (2025)

A integração com a Groq API é um dos diferenciais do projeto, pois permite executar modelos *open source* com latência reduzida e consistência nos parâmetros, dispensando infraestrutura local de alto desempenho. O Google Generative AI API, por sua vez, foi utilizado para execução dos modelos da família Gemini, mantendo a comparabilidade entre as respostas.Os parâmetros utilizados, como temperatura, número máximo de tokens, número de execuções e *timeouts*, foram padronizados, assegurando que as comparações entre modelos fossem conduzidas sob as mesmas condições experimentais.

Por fim, a arquitetura modular e documentada reflete boas práticas de engenharia de software aplicadas à pesquisa científica, promovendo transparência, reprodutibilidade e validação dos resultados.

4.2.1 Estrutura de Diretórios

A estrutura de diretórios do projeto reflete a modularidade do sistema, assegurando a independência entre os componentes e facilitando o versionamento e a manutenção. Essa organização hierárquica é apresentada detalhadamente no Quadro 3, enquanto o projeto completo, incluindo o código-fonte integral e scripts de configuração, encontra-se disponível para consulta no Apêndice A.

Quadro 3 — Estrutura de diretórios e arquivos do *framework*

```
llm-comparison-benchmark/
├── main.py                    # Ponto de entrada principal
├── requirements.txt           # Dependências Python
├── src/                       # Código fonte principal
│   ├── config.py              # Configurações centralizadas
│   ├── pipeline.py            # Pipeline de execução dos LLMs
│   ├── models.py              # Implementação dos modelos
│   ├── utils.py               # Utilitários e funções auxiliares
│   └── logger.py              # Sistema de logging
├── prompts/                   # Prompts e benchmarks
│   ├── prompts.json           # 20 prompts padronizados estruturados
│   └── benchmarks.json        # Benchmarks padronizados
├── analysis/                  # Sistema de análise avançada
│   ├── __init__.py            # Pacote de análise
│   ├── analysis.py            # Orquestrador principal de análise
│   ├── ranking_system.py      # Sistema de ranking comparativo
│   ├── benchmarks.py          # Classe base para benchmarks
│   ├── mmlu.py                # Calculadora MMLU
│   ├── hellaswag.py           # Calculadora HellaSwag
│   ├── bleu_rouge.py          # Cálculo de métricas BLEU/ROUGE
│   ├── bertscore.py           # Cálculo de métricas BERTScore
│   └── evidently_reports.py   # Geração de relatórios Evidently AI
│
├── results/                    # Resultados das execuções
│   ├── resultado_1/           # Execução 1
│   ├── resultado_2/           # Execução 2
│   └── resultado_N/           # Execução N
├── analysis/                  # Análises consolidadas
└── analise_consolidada_YYYYMMDD_HHMMSS/
```

Fonte: Elaborado pelo autor (2026)

4.3 EXECUÇÃO DOS EXPERIMENTOS

Os experimentos foram conduzidos com base no *pipeline* definido no Capítulo 3, respeitando os parâmetros e métricas estabelecidos. O processo de execução seguiu uma sequência estruturada de etapas, descritas a seguir:

1. **Preparação dos insumos:** leitura e validação dos *prompt*s e *benchmark*s armazenados nos arquivos prompts.json e benchmarks.json;
2. **Configuração dos parâmetros de execução:** definição de temperatura, tamanho máximo de tokens, número de execuções e *timeouts*, aplicados de forma uniforme a todos os modelos;
3. **Inicialização do *pipeline*:** carregamento dos modelos definidos em models.py e ativação das integrações Groq e Google Generative AI;
4. **Execução automatizada:** processamento sequencial das solicitações (*prompt*s) para cada modelo, com registro dos tempos de resposta e salvamento das saídas no diretório /*results*;
5. **Análise consolidada:** após a coleta, analysis.py lê os resultados brutos, chama os módulos de métrica (bleu_rouge.py, bertscore.py, mmlu.py, hellaswag.py), agrega os resultados por modelo e execução, aplica as normalizações (ex.: Min-Max) e produz arquivos intermediários;
6. **Ranking e relatórios finais:** com os artefatos consolidados, ranking_system.py gera rankings individuais e consolidados e em seguida, evidently_reports.py utiliza esses dados para criar relatórios HTML de qualidade de dados (EvidentlyAI) e painéis estáticos para inspeção humana;
7. **Armazenamento e versionamento:** todos os artefatos finais (relatório consolidado, relatório normalizados e rankings) são salvos em diretórios datados analysis_consolidated/YYYYMMDD_HHMMSS/ para garantir rastreabilidade e reprodutibilidade.

Durante o processo, foi mantido o controle de logs e erros via módulo logger.py, garantindo rastreabilidade completa de cada execução. As execuções foram realizadas em

ambiente controlado, com rede estável e chaves de API dedicadas, assegurando consistência e repetibilidade.

4.4 EXECUÇÃO DEMONSTRATIVA

Com o objetivo de validar a operacionalidade do *framework* desenvolvido, foi conduzida uma execução demonstrativa composta por:

- 20 *prompt*s padronizados + 7 *prompts de benchmarks*;
- 3 execuções completas;
- 7 modelos de linguagem (4 famílias), *open source* via Groq e proprietários via Google API.

É importante ressaltar que a execução apresentada nesta seção possui caráter de Prova de Conceito (*Proof of Concept* - PoC), com o objetivo de validar a integridade funcional do *framework*, a comunicação via APIs e a aplicação correta das métricas de avaliação. A amostragem utilizada (20 *prompts* + 7 *prompts de benchmarks*) não possui significância estatística suficiente para determinar superioridade cognitiva de um modelo sobre o outro. Portanto, os dados quantitativos devem ser interpretados como uma demonstração da capacidade de execução da ferramenta proposta, e não como um *benchmark* de mercado.

4.4.1 Parâmetros de Execução

Para garantir condições experimentais uniformes entre os modelos, foram adotados os seguintes parâmetros:

- *temperature: 0.2*
- *max_tokens: 256*
- *timeout_entre_perguntas: 3s (+4s para modelos Gemini)*
- *numero_execucoes: 3*
- *timeout_entre_execucoes: 30s*
- *Benchmark*s habilitados.

A escolha de temperatura igual a 0,2 fundamenta-se na necessidade de minimizar a aleatoriedade nas respostas geradas, promovendo maior determinismo e consistência entre execuções. Esses parâmetros foram definidos no módulo de configuração do projeto e aplicados de forma padronizada em todas as chamadas realizadas pelo *pipeline*.

4.4.2 Execução e Coleta dos Resultados

Com os parâmetros definidos, o arquivo main.py foi responsável por executar o *pipeline*, enviando os 27 *prompt*s a cada um dos oito modelos nas três execuções planejadas. Para cada chamada às APIs, o sistema registrou informações essenciais para auditoria e rastreabilidade, incluindo o modelo utilizado, o identificador do *prompt*, o texto retornado, o tempo de resposta, o status da requisição (sucesso, timeout, erro de cota ou resposta vazia) e os metadados adicionais. Ao final de cada execução, os resultados brutos, armazenados em arquivos CSV e JSON, foram organizados em diretórios separados, conforme estrutura:

- */results/resultado_1/*
- */results/resultado_2/*
- */results/resultado_3/*

O Quadro 4 apresenta um trecho ilustrativo do arquivo resultados_todos.csv que consolida as respostas obtidas na execução registrada em /results/resultado_1/.

Quadro 4 — Trecho ilustrativo do arquivo resultados_todos.csv


```
{
       "model":"llama3_70b",
       "prompt":"Quais são os principais desafios éticos da IA? Discuta
       implicações sociais e possíveis soluções.",
       "reference":"Os principais desafios éticos da IA (...)",
       "prediction":"**Desafios Éticos da IA: (...)",
       "time":0.9864778519,
       "timestamp":"2026-03-06T17:22:02.175439",
       "prompt_length":96,
       "response_length":933,
       "is_error":false,
}
```


Fonte: Elaborado pelo autor (2026).

Nota: Trecho reduzido para fins ilustrativos.

Durante essa etapa, o *pipeline* registrou também eventuais falhas de API, respostas truncadas ou conteúdo não processável. Essas ocorrências foram automaticamente registradas pelo módulo logger.py e marcadas como indisponíveis para análises posteriores, garantindo que a execução completa do experimento não fosse interrompida. A seguir, o Quadro 5 demonstra um trecho ilustrativo do arquivo relatório_erros.json que consolida os erros obtidos na execução registrada em /results/resultado_1/.

Quadro 5 — Trecho ilustrativo do arquivo relatorio_erros.csv


```
{
  "resumo": {
    "total_erros": 23,
    "total_respostas": 189,
    "taxa_erro": 12.17,
    "data_analise": "2026-03-06 17:47:35"
  },
  "erros_por_modelo": {
    "gpt_oss_20b": {
      "Other Error": 3
    },
    "gpt_oss_120b": {
      "Other Error": 2
    },
    "gemini_2_5_flash_lite": {
      "Rate Limit": 10
    },
    "gemini_3_flash_preview": {
      "Rate Limit": 8
    }
  },
  "detalhes_erros": [
    {
      "modelo": "gpt_oss_20b",
      "prompt": "Como funciona a fotossíntese e por que é importante para a vida
na Terra? Inclua detalhes sobre o pr...",
      "erro": "[ERRO]: Resposta vazia do modelo gpt_oss_20b",
      "tempo": 0.4768640995025635
    },
(...)
     {
      "modelo": "gemini_2_5_flash_lite",
      "prompt": "Context: A child is playing with a ball in the backyard. The
ball rolls under a bush and the child c...",
      "erro": "[ERRO]: Rate limit ou quota excedida para gemini_2_5_flash_lite",
      "tempo": 38.093040466308594
    },
    {
      "modelo": "gemini_3_flash_preview",
      "prompt": "Describe the impact of 5G technology on modern communication,
including benefits for IoT and smart c...",
      "erro": "[ERRO]: Rate limit ou quota excedida para gemini_3_flash_preview",
      "tempo": 37.80734729766846
```

```
    },
(...)
```

Fonte: Elaborado pelo autor (2026).

Nota: Trecho reduzido para fins ilustrativos.

Nessa etapa, não houve cálculo de métricas ou filtragem de respostas. Todos os dados foram preservados em sua forma original, servindo como base para as fases posteriores de análise. Para fins de registro, contabilizou-se o volume total de dados obtidos nas três execuções. Entre os sete modelos avaliados, foram produzidas 567 respostas, das quais 428 foram classificadas como válidas, resultando em uma taxa global de sucesso de 75,5%.

Esses números representam apenas o volume de dados coletados e ilustram a capacidade do *pipeline* de registrar, de forma estruturada e completa, todas as interações realizadas durante a execução demonstrativa.

4.4.3 Consolidação e cálculo de métricas

Concluída a fase de coleta e tratamento de erros, os dados armazenados em /results/ foram processados pelo módulo analysis/analysis.py. Essa etapa consistiu na leitura e agregação das respostas válidas por modelo e por execução; na organização das informações em estruturas tabulares; e no cálculo das métricas acadêmicas descritas na Seção 3.1, a saber: ROUGE-1, ROUGE-2, ROUGE-L, BLEU e BERTScore.

Para ilustrar esse processo, o Quadro 6 demonstra um trecho ilustrativo do arquivo \analysis\bleu_rouge.py, responsável por realizar o cálculo das métricas de BLEU e ROUGE.

Quadro 6 — Trecho do algoritmo de cálculo das métricas BLEU e ROUGE

```
# Inicialização das métricas
from nltk.translate.bleu_score import sentence_bleu, SmoothingFunction
from rouge_score import rouge_scorer
smoothing = SmoothingFunction().method1
rouge_scorer_instance = rouge_scorer.RougeScorer(
    ['rouge1', 'rouge2', 'rougeL'],
    use_stemmer=True )
(...)
for idx, row in df.iterrows():
    resposta_esperada = str(row.get('reference', ''))
    resposta_modelo = str(row.get('prediction', ''))
    # Preparação das respostas
    reference = [resposta_esperada.split()]
    candidate = resposta_modelo.split()
```

```
    # Cálculo da métrica BLEU
    bleu_score = sentence_bleu(
        reference,
        candidate,
        smoothing_function=smoothing
    )
    # Cálculo das métricas ROUGE
    rouge_scores = rouge_scorer_instance.score(
        resposta_esperada,
        resposta_modelo
    )
    rouge1_score = rouge_scores['rouge1'].fmeasure
    rouge2_score = rouge_scores['rouge2'].fmeasure
    rougeL_score = rouge_scores['rougeL'].fmeasure
(...)
# Cálculo das métricas médias por modelo
metricas_por_modelo[modelo] = {
    'bleu_medio': df_validas['bleu_score'].mean(),
    'rouge1_medio': df_validas['rouge1_score'].mean(),
    'rouge2_medio': df_validas['rouge2_score'].mean(),
    'rougeL_medio': df_validas['rougeL_score'].mean()
}
```

Fonte: Elaborado pelo autor (2026).

Nota: Trecho adaptado e reduzido para fins ilustrativos.

As métricas foram calculadas apenas quando havia quantidade suficiente de respostas válidas para aquele modelo, evitando a geração de estatísticas sobre bases muito reduzidas ou incompletas. Alguns casos nessa execução demonstrativa, como, por exemplo, nos modelos Gemini, o grande número de falhas de API inviabilizou o cálculo consistente das métricas acadêmicas.

4.4.4 Processamento dos benchmarks

Além das métricas acadêmicas textuais, a execução demonstrativa incluiu a aplicação dos *benchmark*s MMLU e HellaSwag. Para cada modelo, o *framework* aplicou o conjunto de questões configurado para MMLU, registrando a acurácia obtida. Na sequência, processou os itens de HellaSwag, identificando a alternativa escolhida pelo modelo em cada instância, e calculou a acurácia global no conjunto de HellaSwag. Por fim, registrou o número total de questões respondidas de forma válida em cada *benchmark*.

Como demonstração técnica dessa etapa, o Quadro 7 ilustra um trecho do arquivo \analysis\hellaswag.py que realiza o cálculo da métrica de *benchmark* HellaSwag.

Quadro 7 — Trecho do algoritmo de cálculo de benchmark HellaSwag

```
class HellaSwagBenchmark(BaseBenchmark):
    """
    Calculadora de métricas para o benchmark HellaSwag.
    """
    def __init__(self):
        super().__init__("hellaswag")
    def calculate_metrics(self, predictions, references):
        if not predictions or not references:
            return {
                "accuracy": 0.0,
                "total_questions": 0,
                "correct_answers": 0
            }
        accuracy = self.calculate_accuracy(predictions, references)
        correct_answers = sum(
            1 for p, r in zip(predictions, references)
            if self.validate_prediction(p, r)
        )
(...)
```

Fonte: Elaborado pelo autor (2026).

Nota: Trecho adaptado e reduzido para fins ilustrativos.

De maneira análoga ao cálculo das métricas textuais, modelos com alta taxa de erro ou respostas incompletas geraram acurácias reduzidas ou não calculáveis. Nesses casos, o *framework* registrou explicitamente as limitações, evitando que os *benchmark*s fossem interpretados como indicadores para modelos cuja execução foi instável na execução demonstrativa.

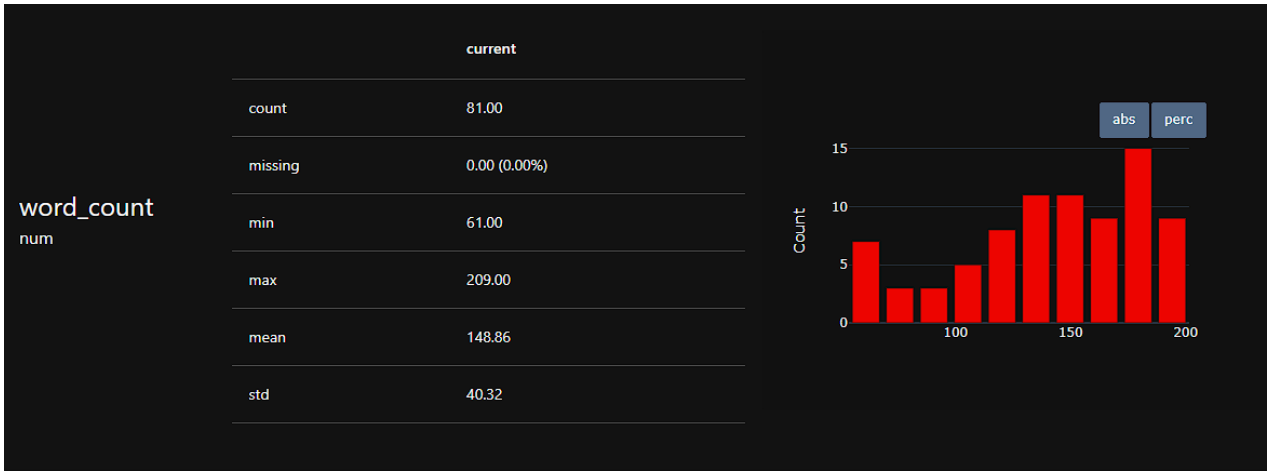
4.4.5 Relatórios de qualidade com EvidentlyAI

Como etapa complementar, o *framework* utilizou a biblioteca EvidentlyAI para gerar relatórios de qualidade textual a partir das respostas válidas coletadas. Entre os aspectos analisados, destacam-se a distribuição do comprimento das respostas por modelo, a variação do tamanho das respostas entre execuções, a identificação de outliers em termos de comprimento

ou estrutura, as possíveis mudanças de distribuição (*drift*) entre diferentes modelos ou execuções e, por fim, os indicadores gerais de consistência e qualidade do texto gerado.

Os relatórios foram produzidos em formato HTML e armazenados junto aos demais artefatos da análise consolidada. A Figura 18, adiante, apresenta a distribuição do número de palavras nas respostas geradas pelo modelo LLaMa 3.3 70B. O gráfico foi produzido automaticamente pela ferramenta EvidentlyAI e permite observar a variabilidade no tamanho das respostas produzidas.

Figura 18 — Trecho do relatório EvidentlyAI



Fonte: Elaborado pelo autor (2026)

4.4.6 Resultados da Execução Demonstrativa

Após a execução demonstrativa, o cálculo de métricas, o *benchmark*, o relatório de qualidade e os resultados consolidados foram armazenados no diretório /analysis/analise_consolidada_20260306_192151/. A Figura 19 ilustra essa dinâmica com um recorte do relatório final em formato *Markdown* gerado automaticamente pelo sistema, que evidencia como os dados são organizados e renderizados diretamente no repositório remoto.

Figura 19 — Trecho do relatório consolidado renderizado no repositório.

Fonte: Elaborado pelo autor (2026)

Em complemento, o Quadro 8 apresenta um resumo das métricas e *benchmark* obtidos a partir das respostas coletadas. Esses resultados refletem exclusivamente esta rodada de execução e foram incluídos com o propósito de ilustrar o correto funcionamento do *framework* de ponta a ponta.

Quadro 8 — Resultados da Execução Demonstrativa

Modelo	Respostas Válidas	Válidas (%)	BLEU	R-1	R-2	R-L	BERT Score	MMLU	HellaSwag
llama-3.1-8b-instant	81	100	0.0394	0.3605	0.1202	0.2062	0.8447	0.9167	1.0000
llama-3.3-70b-versatile	81	100	0.0405	0.3581	0.1247	0.2064	0.8451	1.0000	1.0000
gpt-oss-120b	76	93.8	0.0188	0.2839	0.0889	0.1796	0.8208	0.8333	1.0000
gpt-oss-20b	69	85.2	0.0105	0.2533	0.0803	0.1646	0.8181	1.0000	1.0000
qwen3-32b	81	100	0.0074	0.1577	0.0400	0.0891	0.8101	0.5000	0.8889
Gemini 2.5 Flash Lite	20	24.7	-	-	-	-	-	-	-
Gemini 3.0 Flash	20	24.7	-	-	-	-	-	-	-

Fonte: Elaborado pelo autor (2026).

Nota: R-1 = ROUGE-1; R-2 = ROUGE-2; R-L = ROUGE-L.

Desse modo, observa-se uma discrepância significativa nos resultados de certos modelos, especificamente o Gemini 2.5 Flash Lite e o Gemini 3.0 Flash em comparação aos demais. Estas variações estão associadas principalmente a falhas recorrentes de API, incluindo respostas vazias, interrupções de requisição e erros relacionados a limites de uso, conforme registrado nos logs de execução do sistema. A baixa taxa de respostas inviabilizou o cálculo das métricas acadêmicas e *benchmark*s, uma vez que o volume de dados não atendeu aos critérios mínimos definidos pelo *framework*.

Diante dessas limitações e variações observadas, optou-se por aplicar uma normalização dos *scores* para permitir uma comparação mais equitativa entre os modelos. O Quadro 9 apresenta os mesmos resultados do Quadro 8, agora ajustados via normalização Min–Max, conforme metodologia descrita na Seção 3.1.6. Essa abordagem possibilita avaliar de forma mais clara as diferenças relativas de desempenho, reduzindo o impacto de métricas absolutas naturalmente baixas em tarefas de geração aberta.

Quadro 9 — Resultados Normalizados da Execução Demonstrativa

Modelo	Respostas Válidas	Válidas (%)	BLEU	R-1	R-2	R-L	BERT Score
llama-3.1-8b-instant	81	100	0.9723	1.0000	0.9639	0.9989	0.9995
llama-3.3-70b-versatile	81	100	1.0000	0.9934	1.0000	1.0000	1.0000
gpt-oss-120b	76	93.8	0.4640	0.7873	0.7130	0.8700	0.9712
gpt-oss-20b	69	85.2	0.2594	0.7026	0.6440	0.7972	0.9680
qwen3-32b	81	100	0.1835	0.4374	0.3204	0.4316	0.9585
Gemini 2.5 Flash Lite	20	24.7	-	-	-	-	-
Gemini 3.0 Flash	20	24.7	-	-	-	-	-

Fonte: Elaborado pelo autor (2026).

Nota: R-1 = ROUGE-1; R-2 = ROUGE-2; R-L = ROUGE-L.

Esses cenários evidenciam a capacidade do *framework* de identificar, registrar e isolar automaticamente ocorrências de instabilidade, sem comprometer a execução global do experimento. Destaca-se que os resultados apresentados correspondem apenas ao subconjunto

de respostas válidas gerado em cada execução, o que inclui variações naturais decorrentes de limitações de API, instabilidades momentâneas e diferenças na forma como cada modelo responde aos *prompt*s. Essas diferenças não constituem objeto de comparação nesta seção, mas demonstram a capacidade do *framework* de tratar entradas, identificar quando os dados não atendem aos critérios mínimos para cálculo de métricas e registrar inconsistências.

4.4.7 Considerações sobre a Execução Demonstrativa

A execução demonstrativa permitiu verificar o funcionamento integral do *pipeline*, o tratamento adequado de falhas de API, o cálculo consistente das métricas e a integração com os módulos de *benchmark*s e análise de qualidade.

Esses resultados reforçam que o *framework* é funcional, robusto e reprodutível, atendendo plenamente ao propósito metodológico estabelecido. A concepção modular e a execução automatizada asseguram transparência, escalabilidade e reprodutibilidade, princípios fundamentais para pesquisas envolvendo modelos de linguagem, estabelecendo uma base sólida para a incorporação futura de novos modelos, métricas, *datasets* ou *benchmarks*.

Por fim, destaca-se que os resultados de execução demonstrativa não têm finalidade comparativa entre modelos, servindo exclusivamente para demonstrar o comportamento do sistema em operação e sua adequação para estudos e análises futuras.

5 CONCLUSÃO

O objetivo principal deste trabalho foi desenvolver um *framework* reprodutível para *benchmarking* de modelos de linguagem, capaz de integrar múltiplas APIs, registrar falhas operacionais e gerar métricas e relatórios automatizados. Esse objetivo foi atingido por meio de uma arquitetura modular em Python, a qual permitiu a integração eficaz de APIs distintas (Google Generative AI e Groq) em um fluxo de avaliação unificado. Os testes realizados com modelos *open source* e proprietários demonstraram a capacidade da ferramenta em padronizar a coleta de métricas e consolidar dados de forma estruturada.

Do ponto de vista da engenharia de software, o *framework* demonstrou robustez, viabilidade operacional e os resultados comprovaram que a solução foi capaz de executar *pipeline*s de avaliação completos, tratar exceções de APIs externas e consolidar métricas acadêmicas e *benchmarks* de forma automatizada. Além disso, a arquitetura modular e o código disponibilizado publicamente permitem que pesquisadores incorporem novos modelos, métricas e conjuntos de dados, ampliando o potencial do *framework* como instrumento de apoio a estudos comparativos e análises experimentais em contextos acadêmicos e aplicados.

Apesar dos resultados promissores, o trabalho apresenta limitações inerentes à sua natureza experimental. Inicialmente, observa-se a dependência de APIs de terceiros, o que sujeita a avaliação a latências de rede, custos de utilização e *rate limits* que podem interromper os testes ou impactar a disponibilidade das respostas devido a filtros de segurança. Outro ponto relevante é a rigidez das métricas baseadas em n-gramas, como ROUGE e BLEU, que podem penalizar respostas semanticamente corretas, mas que divergem lexicalmente da referência. Adicionalmente, as execuções não permitem generalização estatística absoluta sobre a capacidade de raciocínio dos modelos, visto que o desempenho pode variar conforme o contexto do *prompt* e a carga nos servidores dos provedores. Ressalta-se, ainda, que as respostas de referências geradas por modelo externo (GPT-5) introduzem uma limitação metodológica, pois os resultados podem refletir a proximidade estilística dos modelos avaliados com o modelo gerador da referência, e não apenas a qualidade intrínseca das respostas.

Para evoluções futuras deste *framework*, sugere-se a implementação de mecanismos de resiliência, tais como políticas de *retry* com e tratamento específico para falhas devido a limite de taxa (*rate limits*) ou instabilidades momentâneas de APIs externas, garantindo maior robustez em execuções de longa duração. Recomenda-se, também, a expansão do suporte para execução de modelos locais eliminando a dependência de internet e APIs externas, e inclusão de um módulo de análise de custo por token processado para cada modelo. Por fim, propõe-se

a implementação da técnica de *LLM-as-a-judge*, utilizando um modelo de maior capacidade para avaliar qualitativamente as respostas de modelos menores, mitigando as limitações das métricas estritamente lexicais.

Em conclusão, o desenvolvimento deste framework não apenas atende à demanda por ferramentas de avaliação independentes e reprodutíveis, mas também estabelece uma base arquitetural sólida para investigações futuras. Espera-se que a solução proposta contribua para a sistematização de *benchmarks* em Modelos de Linguagem, oferecendo à comunidade acadêmica e técnica um instrumento eficaz para acompanhar e mensurar a rápida evolução da Inteligência Artificial Generativa.

6 REFERÊNCIAS

ALLES, V. J. **Construção de um corpus para extrair entidades nomeadas do Diário Oficial da União utilizando aprendizado supervisionado**. 60 f. 2018. Dissertação (Mestrado em Engenharia Elétrica) - Universidade de Brasília, Brasília, 2018.

BASHEER, I. A.; HAJMEER, M. N. **Artificial neural networks: fundamentals, computing, design, and application.** Journal of Microbiological Methods, v. 43, n. 1, p. 3-31, 2000.

BIRD, S.; KLEIN, E.; LOPER, E. **Natural Language Processing with Python**: Analyzing Text with the Natural Language Toolkit. v. 3. O'Reilly Media, Inc., 2009. 506 p.

BUSSLER, F. **Why Open Source Language Models Are True "Open AI"**. Disponível em: https://hackernoon.com/why-open-source-language-models-are-true-open-ai. Acesso em: 11 jun. 2024.

CAMPOS, A. O. G. **Estudo, avaliação e comparação de técnicas de detecção não supervisionada de outliers**. 151 f. 2015. Dissertação (Mestrado em Engenharia) - Universidade de São Paulo, São Carlos, 2015.

CHOWDHURY, G. G. **Natural language processing.** Annual Review of Information Science and Technology, Wiley Online Library, v. 37, n. 1, p. 51-89, 2003.

CICHY, R. M.; KAISER, D. **Deep neural networks as scientific models**. Trends in Cognitive Sciences, Elsevier, v. 23, p. 305-317, 2019.

CLOUDSMITHS. **Open-Source vs. Proprietary LLMs: Which Model is Right for Your Operations?** Disponível em: https://www.cloudsmiths.co.za/blog/open-source-vs-proprietary-llms-which-model-is-right-for-your-operations. Acesso em: 10 set. 2024.

DAMBROS, H. V. **Predição de alvos de microRNAs utilizando aprendizado semi-supervisionado e classificação baseada em uma única classe**. 194 f. 2019. Dissertação (Mestrado em Engenharia) - Universidade Federal do Rio Grande do Sul, Porto Alegre, 2019.

DEEPSEEK. **DeepSeek - A next-generation semantic search platform**. Disponível em: https://www.deepseek.com/en. Acesso em: 20 maio 2025.

DEVLIN, J. et al. **BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding**. In: ANNUAL CONFERENCE OF THE NORTH AMERICAN CHAPTER OF THE ASSOCIATION FOR COMPUTATIONAL LINGUISTICS (NAACL-HLT), 2019, Minneapolis. *Proceedings...* Minneapolis: ACL, 2019. p. 4171–4186. Disponível em: https://arxiv.org/abs/1810.04805. Acesso em: jun. 2025.

DOMINGOS, P. **O Algoritmo Mestre**: Como a busca pelo algoritmo de machine learning definitivo recriará nosso mundo. Novatec Editora, 2017. 27 p.

EISENSTEIN, J. **Introduction to Natural Language Processing**. v. 1. MIT Press, 2019.

EVIDENTLYAI. **Evidently AI — Open-source monitoring for ML models**. Disponível em: https://evidentlyai.com/. Acesso em: 20 maio 2025.

FIREWORKS AI. **DeepSeek v3 and R1 Model Architecture**: why it's powerful and economical. Disponível em: https://fireworks.ai/blog/deepseek-model-architecture. Acesso em: 20 maio 2025.

FUNDAÇÃO LINUX. **2023 Open Source Generative AI Survey Report**. 2023. Disponível em: https://www.linuxfoundation.org/hubfs/LF%20Research/GenAI_Report_2023_020524.pdf?hsLang=en. Acesso em: 23 abr. 2024.

GABRIEL, M. **Inteligência Artificial – Do Zero ao Metaverso**. 1. ed. São Paulo: Atlas, 2022.

GÉRON, A. **Mãos à obra**: **aprendizado de máquina com Scikit-Learn, Keras & TensorFlow: Conceitos, ferramentas e técnicas para a construção de sistemas inteligentes.** 2021.

GOOGLE. **Gemini – Google DeepMind**. Google DeepMind, 05 mar. 2026. Disponível em: https://deepmind.google/models/gemini/. Acesso em: mar. 2026.

H2O.AI. **Open Source GPT**. Disponível em: https://h2o.ai/platform/open-source-gpt/. Acesso em: 08 jul. 2024.

HAYKIN, S. S. **Neural Networks and Learning Machines**. Prentice Hall, 2008. 938 p.

HENDRYCKS, D. et al. **Measuring Massive Multitask Language Understanding**. International Conference on Learning Representations (ICLR), 2021. Disponível em: https://arxiv.org/abs/2009.03300. Acesso em: 16 ago. 2025.

IBA, H.; NOMAN, N. **Deep Neural Evolution**: Deep Learning with Evolutionary Computation. Springer Nature, 2020. 437 p.

INDURKHYA, N.; DAMERAU, F. J. **Handbook of Natural Language Processing**. 2. ed. v. 1. CRC Press, 2010. 704 p.

KNIGHT, W. **Meta Releases Llama 3.2—and Gives Its AI a Voice**. Disponível em: https://www.wired.com/story/meta-releases-new-llama-model-ai-voice/. Acesso em: 10 out. 2024.

LIN, C.-Y. ROUGE: **A package for automatic evaluation of summaries.** In: TEXT SUMMARIZATION BRANCHES OUT: Proceedings of the ACL-04 workshop, Barcelona, Spain, 2004. p. 74–81.

LIU, Y. et al. **RoBERTa: A Robustly Optimized BERT Pretraining Approach**. arXiv preprint arXiv:1907.11692, 2019. Disponível em: https://arxiv.org/abs/1907.11692. Acesso em: 25 maio 2025.

MARSLAND, S. **Machine Learning: An Algorithmic Perspective.** Boca Raton: CRC Press, 2011. 406 p.

MCCULLOCH, W. S.; PITTS, W. **Logical Calculus of the Ideas Immanent in Nervous Activity.** Mind, v. 5, 1943.

META. **Apresentando Meta Llama 3: o grande modelo de linguagem de código aberto mais capaz até hoje**. 2024. Disponível em: https://about.fb.com/br/news/2024/04/apresentando-meta-llama-3-o-grande-modelo-de-linguagem-de-codigo-aberto-mais-capaz-ate-hoje/. Acesso em: 08 jul. 2024.

MELLIT, A.; PAVAN, M.; OGLIARI, E. **Advanced Methods for Photovoltaic Output Power Forecasting: A Review.** Applied Sciences, 2020. 487 p. Disponível em: https://www.researchgate.net/publication/338510612_Advanced_Methods_for_Photovoltaic_Output_Power_Forecasting_A_Review. Acesso em: 3 out. 2023.

MINAEE, S. et al. **Large Language Models: A Survey**. arXiv, 2024.

MIT TECHNOLOGY REVIEW. **Os modelos abertos de IA são mesmo open source?** Disponível em: https://mittechreview.com.br/os-modelos-abertos-de-ia-sao-mesmo-open-source/. Acesso em: 20 jul. 2024.

MITCHELL, T. M. **Machine Learning**. McGraw-Hill Science/Engineering/Math, 1997. 414 p.

MOHAMMED, M.; KHAN, M. B.; BASHIER, E. M. B. **Machine Learning**: **Algorithms and Applications.** v. 1. CRC Press, 2016. 227 p.

MOTTA, E. N. **Indução e Seleção Incrementais de Atributos no Aprendizado Supervisionado**. 90 f. 2014. Dissertação (Mestrado em Engenharia) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2014.

MUELLER, J. P.; MASSARON, L. **Aprendizado de máquina para leigos**. 1. ed. 2019.

NAVEED, H. et al. **A Comprehensive Overview of Large Language Models**. ArXiv, 2024.

NIELSEN, A. **Análise Prática de Séries Temporais**: Predição com Estatística e Aprendizado de Máquina. 1. ed. 2021.

OPARA, E. C.; ADALIKWU, M.-E. T.; TOLORUNLEKE, C. A. **ChatGPT for Teaching, Learning and Research: Prospects and Challenges.** Global Academic Journal of Humanities and Social Sciences, v. 5, n. 2, p. 33-40, 2023. Disponível em: https://doi.org/10.36348/gajhss.2023.v05i02.001. Acesso em: 20 nov. 2024.

OPENAI. **Introducing gpt-oss**. 2025. Disponível em: https://openai.com/index/introducing-gpt-oss/. Acesso em: 30 nov. 2025.

OPEN SOURCE INITIATIVE. **The Open Source Definition**. 2024. Disponível em: https://opensource.org/. Acesso em: 23 abr. 2024.

PAPINENI, K. et al. **BLEU: a method for automatic evaluation of machine translation**. In: ANNUAL MEETING ON ASSOCIATION FOR COMPUTATIONAL LINGUISTICS, 40., 2002, Philadelphia. *Proceedings...* Philadelphia: ACL, 2002. p. 311–318.

PENG, J. et al. **Machine Learning Techniques for Personalised Medicine Approaches in Immune-Mediated Chronic Inflammatory Diseases: Applications and Challenges.** Front. Pharmacol., 2021. Disponível em: https://doi.org/10.3389/fphar.2021.720694. Acesso em: 4 out. 2023.

RAFFEL, C. et al. **Exploring the limits of transfer learning with a unified text-to-text transformer.** The Journal of Machine Learning Research, v. 21, n. 1, p. 5485–5551, 2020.

RAMOS, A. S. M. **Generative Artificial Intelligence based on large language models - tools for use in academic research**. SciELO Preprints, 2023. Disponível em: https://doi.org/10.1590/SciELOPreprints.6105. Acesso em: 15 abr. 2024.

REUTERS. **ChatGPT sets record for fastest-growing user base**. Reuters, 1 fev. 2023. Disponível em: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/. Acesso em: 13 jun. 2024.

RÖHE, A.; SANTAELLA, L**. IAs generativas – a importância dos comandos para texto e imagem.** Aurora – revista de arte, mídia e política, v. 16, n. 47, 2023. Disponível em: https://doi.org/10.23925/1982-6672.2023v16i47p76-94. Acesso em: 16 jun. 2024.

SANTOS, C. **Explorando o GPT-4 e Comparando suas Características com o ChatGPT**. Disponível em: https://updf.com/br/knowledge/what-is-gpt-4/. Acesso em: 03 abr. 2024.

SILVA, I. N. **Redes Neurais Artificiais Para Engenharia e Ciências Aplicadas**: Fundamentos Teóricos e Aspectos Práticos. 2. ed. São Paulo: Artliber, 2016.

SILVERIO, M. **Aplicação de algoritmos de aprendizado de máquina no desenvolvimento de modelos de escore de crédito**. 61 f. 2015. Dissertação (Mestrado Profissional em Ciências) - Insper Instituto de Ensino e Pesquisa, 2015.

SILVESTRE, P. **Open source promete revolucionar a inteligência artificial**. Disponível em: https://itforum.com.br/colunas/open-source-inteligencia-artificial/. Acesso em: 18 jul. 2024.

SUGIYAMA, M. et al. **Machine Learning from Weak Supervision**: An Empirical Risk Minimization Approach. v. 1. MIT Press, 2022. 315 p.

TAULLI, T. **Introdução à Inteligência Artificial**: Uma abordagem não técnica. v. 2. Novatec Editora, 2020. 219 p.

THE GUARDIAN. **China's DeepSeek chatbot shows the West what an AI superpower looks like.** The Guardian, Londres, 28 jan. 2025. Disponível em: https://www.theguardian.com/commentisfree/2025/jan/28/deepseek-r1-ai-world-chinese-chatbot-tech-world-western. Acesso em: 20 maio 2025.

TURING, A. M. **Computing Machinery and Intelligence.** Mind, Volume LIX, Issue 236, p. 433–460, out. 1950. Disponível em: https://doi.org/10.1093/mind/LIX.236.433. Acesso em: 27 jun. 2025.

VASWANI, A. et al. **Attention is all you need**. Advances in neural information processing systems, v. 30, 2017.

VEIGA, A. O. **Treino não supervisionado de modelos acústicos para reconhecimento de fala**. 143 f. 2013. Tese (Doutorado em Engenharia) - Universidade de Coimbra, Coimbra, 2013.

WANG, L. et al. **Open-source large language models: Opportunities and challenges.** Journal of Systems and Software, v. 205, 112452, 2025. Disponível em: https://doi.org/10.1016/j.jss.2025.112452. Acesso em: 20 maio 2025.

WEBSTER, J. J.; KIT, C. **Tokenization as the initial phase in nlp**. In: COLING, volume 4: The 14th international conference on computational linguistics, 1992.

YANG, A. et al. **Qwen3 Technical Report**. 2025. Disponível em: https://arxiv.org/abs/2505.09388. Acesso em: set. 2025.

ZELLERS, R. et al. **HellaSwag: Can a Machine Really Finish Your Sentence?**. 2019. Disponível em: https://arxiv.org/abs/1905.07830. Acesso em: 16 ago. 2025.

ZHANG, T. et al. BERTScore: Evaluating Text Generation with BERT. In: **INTERNATIONAL CONFERENCE ON LEARNING REPRESENTATIONS (ICLR)**, 2020. Disponível em: https://arxiv.org/abs/1904.09675. Acesso em: 29 maio 2025.

ZHANG, C. et al. **100 days after DeepSeek-R1: a survey on replication studies and more directions for reasoning language models**. 2025. Disponível em: https://arxiv.org/abs/2505.00551. Acesso em: 20 maio 2025.

APÊNDICE A – REPOSITÓRIO E ARTEFATOS

Este apêndice disponibiliza o repositório contendo o código-fonte do *framework* desenvolvido, bem como todos os artefatos gerados durante as execuções demonstrativas, incluindo resultados brutos, análises consolidadas e relatórios de qualidade textual.

O repositório pode ser acessado em:
https://github.com/caiocezarq/llm-comparison-benchmark