CSBC26
19 A 23 DE JULHO | GRAMADO

JAI 2026

45ª JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA

EDITORES

Carla M. Dal Sasso Freitas
Raquel da Silva Cabral
Rodrigo Brandão Mansilha

SBC
Sociedade Brasileira de Computação

CSBC26

46º Congresso da
Sociedade Brasileira de Computação

Gramado, RS - 19 a 23 de julho de 2026
https://csbc.sbc.org.br/2026/

45ª Jornada de Atualização em Informática

Editores
Carla M. Dal Sasso Freitas (UFRGS)
Raquel da Silva Cabral (UFAL)
Rodrigo Brandão Mansilha (UNIPAMPA)

Organização
Universidade Federal do Rio Grande do Sul – UFRGS

Realização
Sociedade Brasileira de Computação – SBC

Capa: Leticia Gonçalves Batista

Dados Internacionais de Catalogação na Publicação (CIP)

C749 Congresso da Sociedade Brasileira de Computação (46.: 16 jul.- 23 jul. 2026 : Gramado)

45ª Jornada de Atualização em Informática (JAI 2026) [recurso eletrônico] / editores: Carla M. Dal Sasso Freitas, Raquel da Silva Cabral e Rodrigo Brandão Mansilha. Dados eletrônicos. - Porto Alegre: Sociedade Brasileira de Computação, 2026.

87 p.; il.: PDF; 8,8 MB

Modo de acesso: World Wide Web.
ISBN: 978-85-7669-675-9 (e-book)

1. Computação - Brasil - Congressos. 2. Informática. I. Freitas, Carla M. Dal Sasso. II. Cabral, Raquel da Silva. III. Mansilha, Rodrigo Brandão. IV. Sociedade Brasileira de Computação. IV. Título.

CDU 004(063)

Ficha catalográfica elaborada por Annie Casali – CRB-10/2339

Biblioteca Digital da SBC – SBC OpenLib

Sociedade Brasileira de Computação
Av. Bento Gonçalves, 9500
Setor 4 | Prédio 43.412 | Sala 219 | Bairro
Agronomia Caixa Postal 15012 | CEP 91501-970
Porto Alegre - RS
(51) 99252-6018
sbc@sbc.org.br

Sociedade Brasileira de Computação – SBC

Contato
Av. Bento Gonçalves, 9500
Setor 4 - Prédio 43.412 - Sala 219
Bairro Agronomia
91.509-900 – Porto Alegre RS

CNPJ: 29.532.264/0001-78
`https://www.sbc.org.br`

45ª Jornada de Atualização em Informática – JAI 2026

Coordenação Geral do CSBC 2026
Alberto Egon Schaeffer Filho (UFRGS)
Weverton Cordeiro (UFRGS)

Coordenação da JAI 2026
Carla M. Dal Sasso Freitas (UFRGS)
Raquel da Silva Cabral (UFAL)

Comitê de Programa da JAI 2026
Alexandre Falcão (UNICAMP)
André Almeida (UFAL)
Bernardo Cunha (UFMG)
Dennis Giovani Balreira (UFRGS)
Jeferson Campos Nobre (UFRGS)
João Paulo Papa (UNESP)
Joel Luís Carbonera (UFRGS)
Jurandy Almeida (UFSCar)
Karina Kohl (UFRGS)
Leila Bergamasco (FEI)
Lucas Mello Schnorr (UFRGS)
Luiz Fernando Bittencourt (UNICAMP)
Mara Abel (UFRGS)
Marcelo Soares Pimenta (UFRGS)
Maria Cristina F. de Oliveira (ICMC/USP)
Omar Paranaíba Vilela Neto (UFMG)
Soraia Raupp Musse (PUCRS)
Thiago de Sales (UFAL)

Coordenação de Publicação
Rodrigo Brandão Mansilha (UNIPAMPA)

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Prefácio

Na edição de 2026 da série Jornada de Atualização em Informática (JAI), apresentamos um conjunto diversificado de textos elaborados como material de apoio aos minicursos selecionados para o evento realizado durante o CSBC 2026. Em sua 45ª edição, a JAI reafirma sua importância acadêmica como um espaço dedicado à atualização científica e tecnológica da comunidade brasileira de Computação. Os textos reunidos neste livro oferecem uma visão abrangente e atualizada de temas relevantes da pesquisa e da prática profissional em Computação. Embora abordem tópicos distintos, compartilham um objetivo comum: apresentar conceitos, métodos e ferramentas relacionados a temas contemporâneos e de grande relevância para o desenvolvimento de soluções tecnológicas capazes de gerar impacto positivo na sociedade.

O primeiro capítulo aborda a Computação Sustentável, tema que assume importância crescente diante da expansão das infraestruturas digitais e do aumento do consumo energético associado às tecnologias da informação. Os autores discutem os impactos socioambientais da computação, apresentam métricas e ferramentas para medição do consumo de energia e da pegada de carbono, além de explorarem estratégias para o desenvolvimento de aplicações mais eficientes e ambientalmente responsáveis. Ao enfatizar a necessidade de incorporar a sustentabilidade aos processos de desenvolvimento e à formação dos profissionais da área, o capítulo contribui para consolidar uma visão mais ampla sobre o papel da Computação na construção de um futuro sustentável.

O segundo capítulo introduz modelos de difusão, tecnologia que atualmente sustenta grande parte dos avanços em síntese de imagens e vídeos gerados por inteligência artificial. De forma acessível e tecnicamente rigorosa, o texto conduz o leitor pelos fundamentos probabilísticos e computacionais desses modelos, explicando como processos de difusão e remoção de ruído permitem a geração de conteúdo visual de alta qualidade a partir de descrições textuais. Ao apresentar os princípios que fundamentam sistemas amplamente utilizados na atualidade, o capítulo oferece uma abordagem didática sobre um dos temas mais relevantes da inteligência artificial generativa.

O terceiro capítulo desloca o foco da tecnologia para as pessoas ao apresentar uma abordagem de desenvolvimento de software centrada no usuário. Com base na experiência acumulada pelos autores, o texto descreve o modelo b_thinking, que integra práticas de UX, Design Thinking, Lean UX e metodologias ágeis em uma estrutura orientada à criação de produtos digitais capazes de atender efetivamente às necessidades dos usuários. Mais do que um conjunto de técnicas, o capítulo propõe uma cultura de desenvolvimento fundamentada em empatia, colaboração, validação contínua e aprendizado compartilhado, demonstrando que a excelência tecnológica depende também da compreensão profunda dos contextos humanos de uso.

Encerrando o livro, o quarto capítulo trata da otimização de prompts em modelos de linguagem de larga escala, tema que ganhou enorme relevância com a popularização dos sistemas baseados em inteligência artificial generativa. Os autores apresentam os fundamentos da engenharia e da otimização de prompts, discutem estratégias como zero-shot, few-shot e chain-of-thought, exploram técnicas avançadas de adaptação e avaliação e analisam desafios relacionados à robustez, à segurança e à confiabilidade desses sistemas. O texto oferece uma visão crítica e atual sobre os mecanismos que permitem direcionar o comportamento dos modelos de linguagem de larga escala e ampliar sua efetividade em diferentes aplicações.

Em conjunto, os capítulos demonstram parte da amplitude e da diversidade dos desafios atuais da Computação, cobrindo desde questões de sustentabilidade e impacto socioambiental, passando pelo desenvolvimento de software centrado no usuário e pelos fundamentos da inteligência artificial generativa, até técnicas avançadas para interação com modelos de linguagem de larga escala.

Agradecemos aos autores pela dedicação na elaboração dos capítulos e aos revisores das propostas pelo trabalho criterioso realizado durante o processo de seleção, que buscou identificar, entre as propostas submetidas, aquelas mais alinhadas a critérios como relevância científica, inovação, atualidade do tema, experiência dos autores, representatividade regional e diversidade institucional.

Esperamos que os textos inspirem reflexões, estimulem novas pesquisas e contribuam para a formação de pesquisadores e profissionais comprometidos com a construção de tecnologias inovadoras, éticas, inclusivas e alinhadas às necessidades da sociedade.

Raquel Cabral e Carla Dal Sasso Freitas
Coordenadoras da JAI 2026

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Sumário

Capítulo 1

Computação Sustentável: da Teoria à Prática com Ferramentas para Medida de Consumo de Energia e Pegada de Carbono das Aplicações

Silvana Rossetto (UFRJ), Luiz Paulo Correa da Silva (UFRJ),
Daniel Cordeiro (USP), Alvaro Luiz Fazenda (UNIFESP),
Alessandro Santiago dos Santos (IPT), Emilio Francesquini (UFABC)

Abstract

Sustainable computing *is concerned with the socio-environmental impacts brought about by computing. Its practice focuses on the adoption of techniques that enable the development of computational solutions that are aware of energy consumption and environmentally sustainable. In this chapter, we will present the relevance and current challenges related to the concept and practice of sustainable computing, its elements and scope, associated techniques, tools, and metrics. We hope that this text emphasizes the importance of the topic, provides theoretical and practical foundations for it, and encourages the reader to use the techniques and tools presented.*

Resumo

A área de estudo ***Computação Sustentável*** *preocupa-se com os impactos socioambientais decorrentes da computação. Sua prática volta-se à adoção de técnicas que permitam o desenvolvimento e o monitoramento de soluções computacionais cientes do consumo energético e ambientalmente sustentáveis. Neste capítulo, apresentaremos a relevância e os desafios atuais da prática de computação sustentável, seus elementos e sua abrangência, bem como as técnicas, ferramentas e métricas associadas. Espera-se que este texto chame a atenção para a importância do tema, forneça fundamentação teórica e prática e estimule o leitor a aprofundar o domínio das técnicas e ferramentas apresentadas.*

Este capítulo foi realizado com recursos da FAPESP, processos 23/00811-0 (“EcoSustain - Ciência de Dados e Computação para o Meio Ambiente”) e 24/01115-0 (“CCD - Cidades Carbono Neutro”); e da FAPERJ, processo E-26/210.101/2025 (“Computação Sustentável e Energeticamente Eficiente”).

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

1.1. Introdução

A crescente demanda por aplicações computacionais de alto desempenho e larga escala — que dependem, em particular, de infraestruturas de *data centers* para processamento e armazenamento de dados — tem chamado a atenção para o consumo de energia requerido e suscitado preocupação, tanto pelos custos financeiros quanto pelos impactos socioambientais associados. Dados apontam que *data centers* consumiram entre 1% e 2% da eletricidade global em 2025.[1] Observa-se também o crescente uso das tecnologias de informação e comunicação (TICs) para automatizar atividades em diversas áreas, entre elas, agricultura, indústria, transporte, comércio, comunicação social, e outras — atingindo diretamente tarefas cotidianas da sociedade moderna. [Launikas et al. 2024] chamam a atenção para a automatização das atividades do dia a dia das pessoas, o tempo acumulado de interação dos usuários com essas aplicações e a consequente responsabilidade individual pelo consumo de energia e pelo impacto ambiental das aplicações que desenvolvemos ou escolhemos usar.

Diante desse cenário, a Sociedade Brasileira de Computação aponta, entre seus grandes desafios para a década de 2025 a 2035, a necessidade de "redução do impacto socioambiental no desenvolvimento e uso de aplicações por meio de Computação Sustentável" [Santos and Wagner 2025][2] . Dentro do escopo desse desafio, as seguintes ações são recomendadas: (a) desenvolver algoritmos energeticamente eficientes; (b) incentivar métodos de programação conscientes; (c) criar novas formas de automação inteligente para apoiar o processo de desenvolvimento; (d) desenvolver hardware e software especializados; (e) incorporar sustentabilidade às métricas de desempenho para Computação de Alto Desempenho.

Construir arquiteturas de hardware e software, bem como aplicações computacionais cientes do consumo de energia, é um desafio e uma prática fundamental já conhecidos nos contextos de computação móvel e da Internet das Coisas (IoT). Nesses casos, o foco é minimizar o consumo de energia para reduzir o aquecimento dos dispositivos e maximizar o tempo de carga das pilhas e baterias. No contexto de aplicações que executam em computadores conectados todo o tempo à rede elétrica — desde aplicações cotidianas até aplicações de larga escala e de alto desempenho — a preocupação com o consumo de energia é mais recente. Essa preocupação decorre da necessidade percebida de minimizar os impactos econômicos e socioambientais gerados pelas próprias aplicações computacionais, incluindo a demanda energética tanto para executá-las quanto para manter a infraestrutura de execução necessária. Desse modo, o consumo de energia de um algoritmo ou aplicação computacional torna-se uma métrica relevante, que precisa ser contrabalançada com métricas mais gerais de tempo de execução e de consumo de memória.

A área de estudo **Computação Sustentável** preocupa-se, de forma ampla, com os impactos econômicos e socioambientais negativos decorrentes da própria Computação e volta-se ao estudo de técnicas e novas práticas que nos permitem mitigar esses impactos [Paul et al. 2023]. No seu escopo mais amplo, preocupa-se não apenas com o

[1]Revista Pesquisa FAPESP – Edição 349 – Março de 2025: `https://revistapesquisa.fapesp.br/os-impactos-ambientais-da-computacao/`.

[2]Grandes Desafios da Computação no Brasil 2025-2035: `https://doi.org/10.5753/sbc.17434.6`

consumo de energia das aplicações, mas com todo o ciclo de produção (projeto e desenvolvimento), utilização e descarte; nos contextos de hardware, software, infraestrutura de comunicação e de armazenamento. Sua prática envolve, entre outras ações: otimizar projetos de hardware e software; monitorar e ter ciência do consumo de energia e da pegada de carbono das aplicações; gerenciar o descarte; e usar, de forma combinada, fontes de energia tradicionais e renováveis.

Neste capítulo, nos atemos ao estudo de: (i) técnicas, ferramentas e métodos para medir o consumo de energia e a pegada de carbono das aplicações; e (ii) estratégias que vêm sendo usadas para minimizar essas métricas. Medir o consumo de energia e a pegada de carbono das aplicações não é uma tarefa trivial. Há vários desafios e diferentes alternativas de procedimento. A pegada de carbono (associada às emissões de CO_2) referente à execução de aplicações computacionais deve considerar todo o ciclo de vida das infraestruturas de hardware, software e comunicação necessárias à execução das aplicações, bem como a geração e transmissão de energia elétrica que sustentam essas infraestruturas. Levantar essa métrica requer, portanto, informações adicionais, para além da aplicação em questão. A partir da ciência dessas métricas, busca-se aplicar técnicas de modelagem e desenvolvimento de software para reduzir o consumo de energia e impacto socioambiental negativo das aplicações.

O restante do texto está organizado em mais quatro seções. A Seção 1.2 apresenta conceitos básicos, desafios e relevância atual da área de estudo da Computação Sustentável. A Seção 1.3 introduz as principais métricas de consumo de energia e de impacto ambiental de aplicações computacionais e apresenta exemplos de ferramentas que auxiliam no levantamento dessas métricas, com destaque para a interface RAPL e a ferramenta CodeCarbon. A Seção 1.4 apresenta brevemente abordagens que vêm sendo utilizadas no desenvolvimento e na execução de aplicações computacionais sustentáveis. Por fim, a Seção 1.5 completa o texto com considerações finais, demandas de pesquisa e aprofundamentos na área.

1.2. Computação Sustentável

Em um cenário mundial de evolução tecnológica crescente e de preocupação com questões ambientais e de sustentabilidade, os temas relacionados à visão de Computação Sustentável têm atraído cada vez mais a atenção e suscitado novas questões de pesquisa. [Phung et al. 2018] destacam que a necessidade de balancear requisitos de desempenho, armazenamento e conservação de energia se reflete em trabalhos de pesquisa que desenvolvem algoritmos de escalonamento cientes de energia [Hoffmann 2015], modelos de potência [Möbius et al. 2013, Phung et al. 2018] e técnicas para redução de consumo de energia em *data centers* [Lee and Zomaya 2012].

Construir soluções computacionais cientes do consumo de energia e de seu impacto ambiental e social torna-se cada vez mais necessário. Trata-se de uma área estratégica que deve viabilizar o desenvolvimento de sistemas de computação que lidam com problemas complexos e envolvem grandes volumes de dados, de forma eficiente, robusta e ambiental e socialmente sustentável. [Lottick et al. 2019] reforçam que "a pegada de carbono dos algoritmos deve ser medida e reportada de forma transparente; só assim os cientistas da computação assumirão um papel honesto e ativo na sustentabilidade ambi-

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

ental."

A Computação Sustentável dialoga com diversas áreas do conhecimento, uma vez que os desafios ambientais atuais são de escala global e interdisciplinares. Significativos avanços na tecnologia da informação e comunicação (TIC), ciência de dados e inteligência artificial nas duas últimas décadas trazem oportunidades para utilizar esses conhecimentos e tecnologias em amplo benefício do meio ambiente.

Por outro lado, os avanços da Computação e o crescimento da complexidade dos problemas tratados geram demandas de consumo de energia significativas. Para frear ou reduzir esse consumo de energia, são necessários esforços em diversas direções, desde o desenvolvimento de soluções computacionais (hardware e software) de baixo consumo energético e alto desempenho até as atualizações na formação dos profissionais da área. Torna-se necessário fomentar e criar condições para a cultura de desenvolvimento de hardware e software cientes do consumo de energia e das emissões de CO_2 e, ao mesmo tempo, com suporte adequado para atender às demandas colocadas, sem implicar aumento significativo da carga de trabalho dos programadores nem custo econômico.

1.2.1. Desafios

Entre os desafios enfrentados atualmente para realizar Computação Sustentável, é possível destacar [Lottick et al. 2019, Paul et al. 2023]:

- **falta de padronização**: amplo espectro de dispositivos, configurações, tecnologias e protocolos, dificultando compatibilidade e interoperabilidade das ferramentas;
- **barreiras técnicas**: necessidade de lidar com *trade-offs* impostos entre diferentes métricas de avaliação das aplicações computacionais;
- **dependências externas**: necessidade de combinar informações externas para medir emissões de carbono;
- **recursos limitados**: poucos investimentos e esforços ainda nessa frente.

A falta de padronização implica a ausência de uniformidade nos métodos utilizados para avaliar a efetividade e a eficiência das tecnologias e soluções propostas para a Computação Sustentável, bem como a dificuldade de comparar resultados e tirar conclusões. Outro desafio associado é a ausência de consenso sobre melhores práticas para medir e reportar o consumo de energia dos sistemas de computação, o que dificulta a comparação da acurácia entre as diferentes soluções.

Em relação às barreiras técnicas, é preciso lidar com o *trade-off* entre eficiência energética e outras métricas de desempenho, como capacidade de processamento, vazão de dados, acurácia e disponibilidade, bem como atender às demandas da sociedade e das novas gerações, ao mesmo tempo em que se reduz o impacto socioambiental negativo.

Para reportar a emissão de CO_2 de uma aplicação, é preciso levar em conta o ciclo de vida dos dispositivos, determinar o local onde a aplicação é executada e o *mix* específico de fontes de energia que abastece este local enquanto a aplicação é executada.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

A composição de fontes de energia que alimentam um sistema de computação é, em muitos casos, o fator mais relevante para determinadas emissões de carbono.

Por fim, a Computação Sustentável requer investimento adicional em hardware energeticamente eficiente e fontes de energia renováveis, além de treinamento e aprendizado adicional para o descarte e a implementação de práticas sustentáveis.

1.3. Como medir consumo de energia de aplicações computacionais

O consumo de energia normalmente é reportado por meio de medidas de energia ou de potência consumida. **Energia** é uma grandeza física escalar que mede a capacidade de um sistema de realizar trabalho ou fornecer calor. Sua unidade de medida no Sistema Internacional (SI) é o joule (J), definido como o trabalho realizado para manter uma corrente de 1 ampere através de uma resistência de 1 ohm durante o intervalo de tempo de exatamente 1 segundo. **Potência** é a taxa de energia consumida (ou gerada) em um dado intervalo de tempo (medida em watts). A medida de *1 watt* corresponde a uma taxa de *1 joule por segundo* (gerada ou consumida), ou seja, a potência refere-se à rapidez de geração ou consumo de energia. Calcula-se a *energia* a partir da *potência* multiplicando-se o valor da potência (gerada ou consumida) pelo intervalo de tempo (de geração ou de consumo). Por exemplo, o consumo de uma lâmpada de 100 watts por 5 segundos corresponde a $100 \times 5 = 500$ joules.

Medidas de energia elétrica são normalmente reportadas em quilowatt-hora (kWh) ou megawatt-hora (MWh). Um quilowatt-hora representa energia consumida a uma taxa de 1.000 watts por hora, o que corresponde a $1.000 \times 3.600 = 3.600.000$ joules. Medidas ambientais associadas à energia consumida são normalmente expressas como emissões por unidade de energia. Por exemplo, *quilogramas de CO_2* emitidos por *quilowatt-hora*.

Há diferentes métodos, com diferentes níveis de granularidade, para medir o consumo de energia. [Jay et al. 2023] propõem a seguinte categorização:

- **Dispositivos externos**: medidores externos posicionados entre o ponto de energia e a fonte de alimentação do computador. Fornecem dados sobre o consumo total de energia do computador.
- **Dispositivos internos**: medidores instalados dentro dos computadores. Podem medir o consumo de energia de componentes individuais.
- **Modelos de energia**: modelos matemáticos e computacionais que aproximam o consumo real.
- **Sensores de hardware e interfaces de software**: sensores digitais e circuitos integrados em componentes de hardware que medem consumo de energia em intervalos de tempo definidos.

O uso de medidores externos requer a aquisição e a implantação desses medidores separadamente e não oferece granularidade e acurácia adequadas para análises detalhadas. Estudos indicaram que a acurácia de soluções baseadas em medidores internos também não é adequada [Kavanagh et al. 2016]. Modelos matemáticos e computacionais para

estimar a energia consumida a partir de contadores de desempenho são impactados por elevadas flutuações na carga de trabalho e pela complexidade e variabilidade do hardware e do software [McCullough et al. 2011].

Neste capítulo, focamos nos métodos e ferramentas de medição do consumo de energia que utilizam **sensores de hardware e interfaces de software**. Em particular, nos centramos em soluções construídas com base na interface RAPL (*Running Average Power Limit*) [Weaver et al. 2012]. RAPL é hoje a interface de referência para estimar o consumo de energia dos processadores [Phung et al. 2018, Thamm and Leser 2025, Diamond and Stoico 2026].

1.3.1. Interface RAPL

A interface RAPL (*Running Average Power Limit*) foi introduzida na arquitetura *Sandy Bridge* da Intel em 2011 [Weaver et al. 2012] e evoluiu em arquiteturas posteriores, como *Haswell* e *Skylake*. Existem hoje interfaces semelhantes em arquiteturas de outros fabricantes, como a da AMD. Permite medir, monitorar e definir limites para a energia consumida pela CPU e pela DRAM associada. Ferramentas de monitoramento do consumo de energia, como CodeCarbon[3] e Scaphandre[4], utilizam a interface RAPL.

[Khan et al. 2018] detalham o funcionamento do RAPL e apresentam uma avaliação cuidadosa das experiências de uso da interface.

Foram utilizados tanto *microbenchmarks* personalizados quanto *benchmarks* de nível de aplicação bem conhecidos, com o objetivo de investigar a acurácia, a granularidade e o desempenho das medidas retornadas pela interface RAPL. Os resultados mais relevantes apontados pelos autores foram: (i) boa acurácia, podendo ser afetada por diferentes arquiteturas, configurações de BIOS e outras configurações de desempenho; e (ii) baixa sobrecarga, podendo variar de uma arquitetura para outra. Além disso, observou-se uma correlação entre o consumo de energia e a temperatura. Na arquitetura Haswell, a energia cresceu de 10–12% entre 37 °C e 74 °C, e, em Skylake, de 8–10% entre 23 °C e 32 °C, indicando a necessidade de "pré-aquecer" a CPU antes das medições.

A interface RAPL é executada automaticamente quando o processador é iniciado, então não há sobrecarga de inicialização na tomada de medidas. Além da possibilidade de medir e monitorar o consumo de energia (foco deste texto), é importante destacar que a interface RAPL também permite definir limites para a potência média consumida pelos componentes internos do processador, o que possibilita o controle do aquecimento desses componentes.

A Figura 1.1 destaca os domínios físicos de gestão de energia abarcados pelo RAPL (que podem variar de uma arquitetura para outra). Para cada domínio, é possível informar o consumo de energia, limitar a potência média consumida dentro de um intervalo de tempo definido, monitorar o impacto da limitação de potência e fornecer outras informações, como unidades de medida e potências mínimas e máximas admitidas pelo domínio. Os domínios destacados incluem:

[3]CodeCarbon: `https://docs.codecarbon.io/`

[4]Scaphandre: `https://hubblo-org.github.io/scaphandre-documentation/`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

- **Package**: energia consumida em um *socket* (inclui o consumo de todos os *cores*, da GPU integrada, do cache e do controlador de memória).
- **Power Plane 0**: energia consumida apenas pelos *cores* de um *socket*.
- **Power Plane 1**: energia consumida apenas pela GPU embutida no *socket*.
- **DRAM**: energia consumida apenas pela RAM associada ao controlador de memória do *socket*.
- **PSys**: introduzido na arquitetura Skylake (outros que não CPU e GPU embutidas).

O único domínio disponível em todas as arquiteturas Intel é o domínio *Package*. Em sistemas com mais de um *socket*, o RAPL reporta valores distintos para cada um deles.

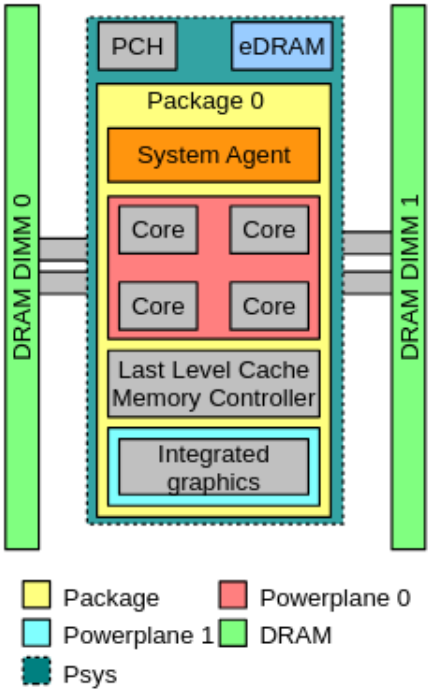

Figura 1.1. Domínios de potência do RAPL (extraído de [Khan et al. 2018])

A forma mais básica de acessar os contadores da interface RAPL é por meio dos MSRs (*Model-Specific Registers*). Trata-se de registradores de 32 ou 64 bits que são atualizados aproximadamente a cada 1 ms e indicam a energia consumida desde o instante em que o sistema foi ligado. Cada arquitetura adota uma unidade de medida básica, e o valor da energia consumida deve ser multiplicado por essa unidade (na arquitetura Sandy Bridge, por exemplo, a unidade básica é de 15,3 microjoules; já nas arquiteturas Haswell e Skylake, a unidade básica é de 61 microjoules). O Código 1.1 mostra um exemplo de acesso à interface RAPL por meio de registrador MSR[5]

[5]Um exemplo mais completo pode ser encontrado aqui: `https://web.eece.maine.edu/~vweaver/projects/rapl/rapl-read.c`, escrito por Vince Weaver.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Código 1.1. Exemplo de código para acesso à interface RAPL (adaptado de [Khan et al. 2018]).

```
uint64_t msr_value;
/* Haswell: units of 61 microjoules  */
/* MSR_PKG_ENERGY_STATUS is at address 0x611 */
/* Package domain */
double energy_units = pow (0.5, 14);
int fd = open ("/dev/cpu/0/msr", O_RDONLY);
if (fd > 0) {
   if (pread (fd, &msr_value, 8, 0x611) > 0) {
      double energy = msr_value * energy_units;
      printf ("%f\n", energy);
}}
```

No exemplo, o modelo da CPU (Haswell) e sua unidade de medida (61 microjoules) foram detectados previamente. O código acessa o dispositivo `/dev/cpu/0/msr` utilizando o endereço do registrador `MSR_PKG_ENERGY_STATUS` (0x611) que informa o consumo de energia do domínio *Package*.

Entre os desafios e limitações de uso do RAPL, [Khan et al. 2018] destacaram a necessidade de acesso com privilégios de administrador do sistema (*root*) aos registradores MSR e o fato de a interface não permitir (na versão avaliada) medir a energia consumida por *core* (mesmo tendo registradores MSR distintos, todos registram o mesmo valor). Além disso, há a possibilidade de estouro dos contadores (limitados a 32 ou 64 bits). Quando eles atingem o limite máximo, retornam a zero. Ao calcular a diferença entre duas medições consecutivas ($E2 - E1$) nesse intervalo, será retornado um valor negativo, pois o primeiro valor será menor que o segundo. Nesses casos, é necessário aplicar correções a esses valores. Por fim, observaram-se também atrasos nas atualizações dos contadores, o que indica que tais atualizações não são atômicas.

Como se observa, o acesso direto aos registradores MSR não é trivial, pois eles fornecem dados brutos que precisam ser tratados. As leituras dos registradores RAPL também podem ser feitas por meio da interface *sysfs powercap* (versões do Linux a partir de 3.13) e com auxílio do `perf_event_open` (versões do Linux a partir de 3.14). [Diamond and Stoico 2026] chamam a atenção para a sobrecarga que pode ser causada pela leitura de energia e investigam estratégias de mitigação. [Kennes 2023] listam diversos trabalhos que reportam o uso do RAPL para a medida do consumo de energia. Há várias ferramentas de medição de energia que usam o RAPL como base. Uma lista extensa está disponível no GitHub[6].

Powercap Trata-se de um *framework* do sistema operacional Linux que provê uma interface de acesso no espaço do usuário, via `sysfs`, na forma de uma *árvore de objetos de controle*. Esses objetos permitem que ferramentas possam monitorar e limitar o consumo de energia de dispositivos de hardware de forma padronizada. Um objeto é sempre

[6] `https://github.com/Green-Software-Foundation/awesome-green-software`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

carregado na memória e, para facilitar a visualização do usuário, aparece virtualmente como um diretório no gerenciador de arquivos. No contexto do powercap, o objeto no topo (raiz) da árvore representa os tipos de controle. Dentro de cada objeto de tipo de controle, temos acesso aos domínios de potência, que contêm atributos de monitoramento e de restrição (fornecendo opções para limitar o consumo de energia de um componente).

Para exemplificar, o Código 1.2 realiza a leitura do consumo de energia via *powercap* em um computador com processador Intel(R) Core(TM) i5-7300HQ CPU 2.50GHz e arquitetura Kaby Lake, utilizando linguagem Python. O domínio de energia consultado foi o *Package*.

Código 1.2. Exemplo de código usando a interface powercap.

```
STRESS_FUNC = ["stress-ng", "--matrix", "0", "-t", "1m", "-v"]
TAXA_AMOSTRAGEM = 1.0

def leitor_powercap():
    '''Lê o contador de energia e o momento da leitura.'''
    caminho_contador = "/sys/class/powercap/intel-rapl/subsystem
       /intel-rapl:0/energy_uj"
    with open(caminho_contador, "r") as powercap:
        valor = powercap.readline().strip()
        tempo_medicao = time.perf_counter()
        return valor, tempo_medicao

def loop_leitor_powercap(duracao, resultados):
    '''Realiza leituras contínuas durante o período de tempo
       dado.'''
    tempo_inicio = time.perf_counter()
    tempo = tempo_inicio
    while (tempo - tempo_inicio <= duracao):
        leitura = [0] * 2
        leitura[0], leitura[1] = leitor_powercap()
        time.sleep(TAXA_AMOSTRAGEM)
        tempo = time.perf_counter()
        resultados.append([leitura[0], leitura[1]])

resultados = []
loop_leitor_powercap(10, resultados)
#inicia stress-ng
stress = subprocess.Popen(STRESS_FUNC)
while(stress.poll() == None):  #enquanto o processo não terminar
    leitura = [0] * 2
    leitura [0], leitura[1] = leitor_powercap()
    resultados.append([leitura[0], leitura[1]])
    time.sleep(TAXA_AMOSTRAGEM)

#10 segundos de execução sem stress-ng
loop_leitor_powercap(10, resultados)
#salva leituras realizadas (...)
```

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Para estressar a CPU, utilizou-se a função `stress-ng -matrix 0` [7] por 1 minuto. Essa função utiliza operações de ponto flutuante e de memória que exigem alto uso da CPU. Note que, para reduzir a sobrecarga da própria medição, o Código 1.2 apenas realiza a leitura dos contadores e faz a persistência dos dados. Para obter os valores de energia consumida, é necessária a execução de outro código que realiza o tratamento e processamento dos dados do RAPL.

Como saída, obtemos um arquivo com diversas linhas e duas colunas (mostradas abaixo). Cada linha representa uma medição, sendo a primeira coluna o valor do contador de energia do RAPL (em microjoules) e a segunda, o valor do contador de tempo (em segundos).

```
37366890973 1108197.640825755
37370550815 1108198.641376323
37374133508 1108199.642308569
37377637650 1108200.643097468
...
```

Para obtermos a potência média consumida, calculamos a diferença entre duas medidas consecutivas (Δe na coluna de energia e Δt na coluna do tempo). A potência média em watts será dada por:

$$p = \Delta e \times 10^{-6} / \Delta t$$

Usando um código auxiliar, processamos os dados coletados para calcular a potência média consumida em cada intervalo de tempo. Como saída, obtemos um arquivo de texto (mostrado abaixo) contendo os valores de potência média (em watts) e o tempo (em segundos) que a medição foi realizada.

```
3.657828116509902 1.0005505681037903
3.579356159605825 2.0014828140847385
3.50137976533455 3.0022717129904777
3.507390855194336 4.0031550319399685
...
```

Com base nesses dados de saída, plotamos o gráfico da Figura 1.2. Podemos observar que a potência média consumida varia claramente ao longo do intervalo de tempo em que o código de estresse foi executado.

1.3.2. Ferramenta Scaphandre

O Scaphandre[8] é um agente de monitoramento escrito na linguagem Rust que permite obter o consumo de energia por processo, por máquina virtual ou por contêiner em execução em uma CPU. A ferramenta utiliza a interface RAPL e é compatível com os sistemas operacionais Windows e GNU/Linux, com algumas funcionalidades restritas a cada sistema operacional.

[7] `https://wiki.ubuntu.com/Kernel/Reference/stress-ng`
[8] `https://github.com/hubblo-org/scaphandre`

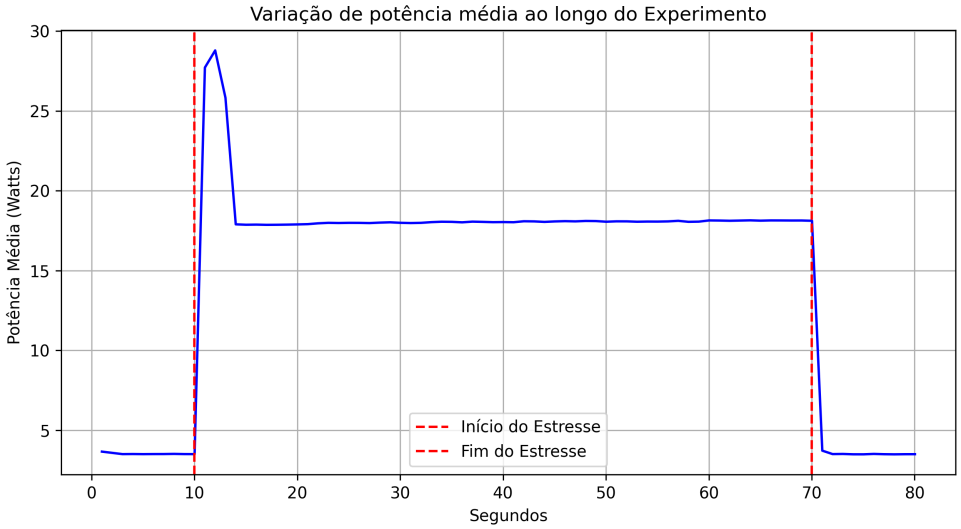

Figura 1.2. Resultado do exemplo usando Powercap.

Para calcular o consumo de energia das aplicações, o Scaphandre coleta continuamente dados dos sensores de hardware e os associa ao tempo de CPU dessas aplicações. A interface de hardware utilizada depende do ambiente no qual a ferramenta será executada. A cada coleta, são lidos os valores do tempo de uso da CPU dos processos ativos no intervalo de medição. Em seguida, é feito o cálculo da estimativa para o consumo de energia por meio da relação entre o consumo de energia total do processador no intervalo medido e a fatia de tempo de CPU que o processo alvo utilizou. Quando programas são executados em processos diferentes ao mesmo tempo, é necessário agregar o consumo de energia de cada processo para obter o gasto total da aplicação.

Em ambiente Linux, o Scaphandre coleta os dados do RAPL via interface *sysfs powercap*, priorizando o domínio PSYS por contemplar a maioria dos componentes. Se esse domínio não estiver disponível na arquitetura-alvo, os dados dos domínios Package e DRAM são somados. Em ambiente Windows, a leitura é feita diretamente dos registradores MSR, via instruções RDMSR.

O Scaphandre foi projetado para ser extensível, limitando-se às tarefas de coleta e pré-processamento das métricas de consumo de energia para, em seguida, exportá-las para outras ferramentas que permitam, por exemplo, visualizar os dados levantados.

1.3.3. Ferramenta CodeCarbon

CodeCarbon[9] é uma biblioteca em Python que permite estimar as emissões de carbono de uma aplicação. Para realizar as medições de consumo de energia, utiliza a interface RAPL.

O CodeCarbon faz estimativas do consumo de energia dos principais componentes do hardware que podem ser monitorados ou estimados diretamente: CPU, GPU e RAM. Ele não modela separadamente E/S de disco, transferências de rede, monitores,

[9] `https://codecarbon.io`

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

resfriamento nem outros periféricos.

Para estimar as emissões de CO_2 associadas à execução de uma aplicação, o CodeCarbon considera que as emissões, expressas em quilogramas de CO_2-equivalente (CO_2eq), são o produto de dois fatores principais:

- C = a intensidade de carbono da eletricidade utilizada para a computação; quantificada em g de CO_2 emitida por quilowatt-hora de eletricidade;
- E = energia consumida pela infraestrutura computacional: quantificada em quilowatt-hora.

As emissões de dióxido de carbono (CO_2eq) são calculadas como o produto $C \times E$.

Intensidade de carbono: A intensidade de carbono da eletricidade consumida (C) é calculada como a média ponderada das emissões de diferentes fontes de energia utilizadas para gerar eletricidade, incluindo combustíveis fósseis e renováveis. No CodeCarbon, os combustíveis fósseis, como carvão, petróleo e gás natural, são associados a intensidades de carbono específicas: uma quantidade conhecida de dióxido de carbono é emitida por quilowatt-hora de eletricidade gerada. As fontes renováveis ou de baixo carbono incluem energia solar, hidroelétrica, biomassa e geotérmica, entre outras. A rede elétrica local contém uma mistura de combustíveis fósseis e de fontes de energia de baixo carbono, chamada Matriz Energética. Com base na proporção de fontes de energia na rede local, a intensidade de carbono da eletricidade consumida pode ser calculada.

Quando disponível, o CodeCarbon usa a intensidade de carbono da eletricidade por país (com dados do site Our World in Data[10]) ou por provedor de computação em nuvem. Se esses dados não estiverem disponíveis, mas detalhes sobre a matriz energética forem conhecidos, o CodeCarbon é capaz de estimar as emissões totais com base na proporção de cada tipo de fonte de energia (carvão, petróleo, gás natural etc.) utilizada para produzir a eletricidade. Em último caso, se nenhuma dessas fontes estiver disponível para o CodeCarbon, utiliza-se a média global de intensidade de carbono: 475 gCO_2eq/kWh.

Consumo de energia: O consumo de energia do hardware subjacente é monitorado em intervalos regulares. Esse comportamento é controlado por um parâmetro configurável (`measure_power_secs`), cujo valor padrão é 15 segundos e pode ser definido no momento da criação do rastreador de emissões (*emissions tracker*).

O CodeCarbon oferece dois modos distintos para determinar o consumo de energia atribuído ao seu trabalho. No **Machine Mode** (`tracking_mode="machine"`), o CodeCarbon mede a energia total consumida por toda a pilha de hardware (todas as CPUs, GPUs e RAM). No **Process Mode** (`tracking_mode="process"`), o CodeCarbon estima a energia do seu código Python (e de seus subprocessos) por meio da amostragem do tempo de CPU em relação à capacidade total da CPU. Esta é uma aproximação baseada em software — ela não lê os contadores de hardware diretamente — e é preferível em ambientes compartilhados onde outras tarefas são executadas em paralelo.

[10] `https://ourworldindata.org/grapher/carbon-intensity-electricity`

A energia consumida por cada tipo de hardware é medida por meio de uma estratégia diferente. A energia consumida pelas GPUs é medida pela biblioteca `nvidia-ml-py`. A energia consumida pela RAM é estimada em 5 W por cada *slot* de RAM utilizado. Já a energia consumida pela CPU depende do sistema operacional e do hardware utilizado.

Atualmente, o CodeCarbon permite medir o consumo de energia de CPUs em ambientes Linux, Windows e macOS. A técnica utilizada para interagir com o hardware e com os domínios de energia consultados depende do ambiente em que se trabalha. No ambiente Linux, os dados de consumo de energia são obtidos via interface RAPL, com o uso do *sysfs powercap*. No ambiente Windows, utiliza a ferramenta Intel Power Gadget[11] (descontinuada pela Intel em 2023). No ambiente macOS, a ferramenta *powermetrics*, nativa desse ambiente, é utilizada. Caso não seja possível obter as medidas de consumo de energia da CPU, o Codecarbon realiza aproximações com base no tipo da CPU-alvo.

O Código 1.3 apresenta um exemplo simples de como o CodeCarbon pode ser integrado a um código em Python para rastrear as emissões. O Código 1.4 mostra como os resultados do rastreamento podem ser acessados programaticamente.

Código 1.3. Exemplo de rastreamento de emissões usando o CodeCarbon.

```python
from codecarbon import EmissionsTracker

# Inicializa o rastreador de emissões
with EmissionsTracker(project_name="meu-primeiro-rastreamento")
    as tracker:
     # Simula alguma computação
     total = 0
     for i in range(10_000_000):
         total += i

     # Imprime o resultado do cálculo
     print(f"Resultado da computação: {total}")
```

1.3.4. Boas práticas para medir consumo de energia

A partir dos relatos feitos por diversos autores, é possível elencar uma lista de recomendações e boas práticas a seguir ao medir o consumo de energia das aplicações[12]:

- **Taxa de amostragem**:[13] A frequência de coleta das medidas de energia deve ser de pelo menos a metade da duração do menor evento que se deseja capturar. Isso garante a precisão necessária para não perder picos de consumo rápidos durante a execução do experimento.
- **Controle da temperatura** [Diamond and Stoico 2026, Khan et al. 2018]: A temperatura de um processador influencia a medição de energia. Sugere-se esperar 180

[11] `https://www.intel.com/content/www/us/en/developer/archive/tools/power-gadget.html`
[12] `https://docs.green-coding.io/docs/measuring/best-practices/`
[13] `https://docs.codecarbon.io/`

Código 1.4. Exemplo de inspeção do resultado do rastreamento do CodeCarbon.

```
import pandas as pd

# Carrega o relatório de emissões gerado pelo CodeCarbon
df = pd.read_csv("emissions.csv")

# Seleciona e exibe colunas específicas do relatório para
    análise
df[["project_name", "duration", "emissions", "emissions_rate", "
    cpu_power", "ram_power", "energy_consumed"]]

#####
# Você também pode acessar os dados de emissões do objeto
    tracker

print(f"Emissões totais: {tracker.final_emissions * 1000:.4f} g
    CO2eq")
print(f"Duração: {tracker.final_emissions_data.duration:.2f}
    segundos")
print(f"Energia consumida: {tracker.final_emissions_data.
    energy_consumed:.6f} kWh")
```

segundos entre as medições para que os componentes esfriem. Em processadores com mais de 30 cores, esse tempo deve ser maior. Se um sistema sobreaquecer durante a medição, isso deve ser considerado.

- **Minimização do ruído de fundo**:[14] Durante a execução do experimento, o gasto energético total da máquina será capturado por meio de interfaces de hardware, como o RAPL. Para minimizar o possível "ruído" do sistema operacional, é importante diminuir a possibilidade de interrupções na CPU. Recomenda-se tomar as seguintes precauções: (a) desativar conexões de rede (Wi-Fi e Internet), a menos que sejam objeto de teste; (b) evitar qualquer interação com periféricos (mouse e teclado) durante a execução; (c) desligar o escurecimento automático da tela, proteção de tela e modos de suspensão; (d) encerrar processos em segundo plano que não sejam essenciais; (e) desativar tarefas agendadas, atualizações automáticas do sistema operacional e rotinas de manutenção.

- **Escrita de resultados na memória**: Para armazenar os valores das medições, é recomendado realizar a escrita em um arquivo que esteja localizado na memória RAM. A escrita em disco é uma operação que gasta bastante energia e, caso seja feita com frequência, pode alterar o experimento a depender das métricas utilizadas.

Para ferramentas que utilizam *Modelos de Divisão de Potência* para estimar o consumo de energia por aplicação isolada, como é o caso do *Scaphandre*, algumas recomendações adicionais incluem manter a frequência de clock fixa, desativar *Hyperthreading* e

[14] https://www.kernel.org/doc/html/next/power/powercap/powercap.html

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Turbo Boost [Cadorel and Saingre 2024]. Além disso, salienta-se que a execução de instruções como AVX (*Advanced Vector Extensions*) pode distorcer o fatiamento de energia, uma vez que o tempo de uso de CPU deixa de ter uma relação clara com o consumo de energia (e, consequentemente, com a potência atribuída a cada processo).

1.4. Abordagens para Computação Sustentável

Algumas abordagens e técnicas têm sido propostas e experimentadas com o objetivo de aplicar os conceitos relacionados à Computação Sustentável, incluindo soluções cientes do consumo de energia, no desenvolvimento e na execução de aplicações computacionais [Cordeiro et al. 2023, Lee and Zomaya 2012, Jones et al. 2018, Almeida et al. 2024].

O conceito de *carbon-responsive computing* engloba estratégias que são cientes da intensidade de CO_2 gerada e usam essa informação para tomar decisões [Nafus et al. 2021]. *Follow-the-renewables* é um exemplo de abordagem nessa direção, que visa incorporar informações sobre a disponibilidade de energia renovável nas decisões de escalonamento de tarefas [Vasconcelos et al. 2023].

Técnicas de otimização já conhecidas e aplicadas em contextos como aprendizado de máquina ou em dispositivos com recursos limitados podem ser usadas para melhorar a eficiência energética, incluindo *pruning* (remover partes desnecessárias de modelos durante o treinamento) e *quantização* (converter dados contínuos ou de alta precisão em conjuntos menores, discretos e finitos).

Outras abordagens envolvem *ciência do hardware* [Carro and Nazar 2023], visando tirar proveito das características específicas do hardware subjacente, e técnicas de *computação aproximada*, que fazem uso de aritmética simplificada ou aproximada para reduzir a complexidade das aplicações [Hoffmann 2015, Rocha et al. 2022].

1.5. Considerações finais

Há vários desafios em aberto a serem enfrentados no contexto de Computação Sustentável [Kennes 2023, Lottick et al. 2019]. Para além de medir o consumo de energia de uma aplicação, há outras demandas a explorar.

Com relação ao *mix* de provisão de energia instantânea, com fontes de energia tradicionais e renováveis, um desafio é a dificuldade de comparar os mesmos intervalos de tempo entre a energia gerada e a energia consumida e a discrepância de granularidade entre a produção medida em horas e o consumo medido em instantes com RAPL. [Kennes 2023] argumenta que a redução do consumo de energia deveria ser realizada para reduzir a produção de energia, facilitando a operação do sistema e evitando emissões decorrentes de requisitos adicionais de energia acima do topo do *baseline* dado.

Como argumenta [Lottick et al. 2019], o cálculo da emissão de carbono é um campo científico próprio. Envolve atribuir o carbono e outros poluentes emitidos na geração, transporte, manutenção, disponibilização e descarte dos recursos, tornando o usuário responsável por essas emissões. Nesse sentido, há várias questões. (a) Como calcular o tempo durante o qual o recurso foi usado por determinada aplicação e sua proporção no tempo total de vida do recurso? (b) Como medir a emissão de poluentes considerando que uma aplicação pode usar milhares de componentes ou recursos? (c) Como tratar

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

o compartilhamento de recursos entre aplicações? (d) Como garantir a transparência e a auditabilidade da emissão de poluentes por aplicações? (e) Quem é responsável pela emissão de poluentes quando o recurso está em sua fase útil, mas não está em uso?

Essas e outras questões estão abertas à pesquisa e à exploração. Esperamos que este texto introdutório desperte o interesse e chame a atenção dos profissionais e pesquisadores da área para as demandas colocadas. Existem ferramentas disponíveis e em desenvolvimento que apoiam o desenvolvimento de aplicações cientes do consumo de energia e da pegada de carbono. Incentivamos o leitor a aprofundar o conhecimento das técnicas e ferramentas apresentadas e a contribuir para a sua evolução e adoção.

Referências

[Almeida et al. 2024] Almeida, G., Torres Do Ó, M., Francesquini, E., and Cordeiro, D. (2024). Reducing carbon emissions of distributed systems: a multi-objective approach. In *Proceedings of the 20th Brazilian Symposium on Information Systems*, pages 1–10.

[Cadorel and Saingre 2024] Cadorel, E. and Saingre, D. (2024). A protocol to assess the accuracy of process-level power models. In *2024 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 74–84. IEEE.

[Carro and Nazar 2023] Carro, L. and Nazar, G. L. (2023). Desafios para a computação energeticamente eficiente. *Sociedade Brasileira de Computação*.

[Cordeiro et al. 2023] Cordeiro, D., Francesquini, E., Amarís, M., Castro, M., Baldassin, A., and Lima, J. (2023). Green cloud computing: Challenges and opportunities. In *Anais Estendidos do XIX Simpósio Brasileiro de Sistemas de Informação*, pages 129–131, Porto Alegre, RS, Brasil. SBC.

[Diamond and Stoico 2026] Diamond, J. and Stoico, V. (2026). What is the cost of energy monitoring? an empirical study on the overhead of rapl-based tools. *arXiv preprint arXiv:2604.26815*.

[Hoffmann 2015] Hoffmann, H. (2015). Jouleguard: Energy guarantees for approximate applications. In *Proceedings of the 25th Symposium on Operating Systems Principles*, pages 198–214.

[Jay et al. 2023] Jay, M., Ostapenco, V., Lefèvre, L., Trystram, D., Orgerie, A.-C., and Fichel, B. (2023). An experimental comparison of software-based power meters: focus on CPU and GPU. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 106–118. IEEE.

[Jones et al. 2018] Jones, N. et al. (2018). How to stop data centres from gobbling up the worlds electricity. *Nature*, 561(7722):163–166.

[Kavanagh et al. 2016] Kavanagh, R., Armstrong, D., and Djemame, K. (2016). Accuracy of energy model calibration with IPMI. In *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, pages 648–655. IEEE.

[Kennes 2023] Kennes, T. (2023). Measuring IT carbon footprint: What is the current status actually? *arXiv preprint arXiv:2306.10049*.

[Khan et al. 2018] Khan, K. N., Hirki, M., Niemi, T., Nurminen, J. K., and Ou, Z. (2018). RAPL in action: Experiences in using rapl for power measurements. *ACM Transactions on Modeling and Performance Evaluation of Computing Systems (TOMPECS)*, 3(2):1–26.

[Launikas et al. 2024] Launikas, A. E., Nakano, F., Coutinho, F. L., Cordeiro, D., et al. (2024). Evolução histórica do desempenho energético de tarefas cotidianas em uma distribuição Linux. In *Simpósio em Sistemas Computacionais de Alto Desempenho (SSCAD)*, pages 81–88. SBC.

[Lee and Zomaya 2012] Lee, Y. C. and Zomaya, A. Y. (2012). Energy efficient utilization of resources in cloud computing systems. *The Journal of Supercomputing*, 60(2):268–280.

[Lottick et al. 2019] Lottick, K., Susai, S., Friedler, S. A., and Wilson, J. P. (2019). Energy usage reports: Environmental awareness as part of algorithmic accountability. *arXiv preprint arXiv:1911.08354*.

[McCullough et al. 2011] McCullough, J. C., Agarwal, Y., Chandrashekar, J., Kuppuswamy, S., Snoeren, A. C., Gupta, R. K., et al. (2011). Evaluating the effectiveness of model-based power characterization. In *USENIX Annual Technical Conf*, volume 20, pages 19–20.

[Möbius et al. 2013] Möbius, C., Dargie, W., and Schill, A. (2013). Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1600–1614.

[Nafus et al. 2021] Nafus, D., Schooler, E. M., and Burch, K. A. (2021). Carbon-responsive computing: Changing the nexus between energy and computing. *Energies*, 14(21):6917.

[Paul et al. 2023] Paul, S. G., Saha, A., Arefin, M. S., Bhuiyan, T., Biswas, A. A., Reza, A. W., Alotaibi, N. M., Alyami, S. A., and Moni, M. A. (2023). A comprehensive review of green computing: Past, present, and future research. *IEEE access*, 11:87445–87494.

[Phung et al. 2018] Phung, J., Lee, Y. C., and Zomaya, A. Y. (2018). Modeling system-level power consumption profiles using RAPL. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)*, pages 1–4. IEEE.

[Rocha et al. 2022] Rocha, F. W., Fukuda, J. C., Francesquini, E., and Cordeiro, D. (2022). Accelerating smart city simulations. In Gitler, I., Barrios Hernández, C. J., and Meneses, E., editors, *High Performance Computing*, pages 148–162, Cham. Springer International Publishing.

[Santos and Wagner 2025] Santos, A. L. d. M. and Wagner, F. R., editors (2025). *Grandes Desafios da Computação no Brasil 2025-2035*. Sociedade Brasileira de Computação (SBC), Porto Alegre.

[Thamm and Leser 2025] Thamm, P. and Leser, U. (2025). Strategies to measure energy consumption using RAPL during workflow execution on commodity clusters. *arXiv preprint arXiv:2505.09375*.

[Vasconcelos et al. 2023] Vasconcelos, M., Cordeiro, D., Da Costa, G., Dufossé, F., Nicod, J.-M., and Rehn-Sonigo, V. (2023). Optimal sizing of a globally distributed low carbon cloud federation. In *2023 IEEE/ACM 23rd International Symposium on Cluster, Cloud and Internet Computing (CCGrid)*, pages 203–215. IEEE.

[Weaver et al. 2012] Weaver, V. M., Johnson, M., Kasichayanula, K., Ralph, J., Luszczek, P., Terpstra, D., and Moore, S. (2012). Measuring energy and power with PAPI. In *2012 41st international conference on parallel processing workshops*, pages 262–268. IEEE.

Capítulo 2

Introdução a Modelos de Difusão para Síntese de Imagens

Manuel Menezes de Oliveira Neto (UFRGS)

Abstract

Diffusion models have emerged as the dominant paradigm for high-quality image and video synthesis. These models generate visual content by reversing a gradual stochastic process that transforms training data into Gaussian noise while conditioning the generation process on guiding inputs such as text prompts. This tutorial provides a gentle yet technically rigorous introduction to diffusion models for image synthesis. Its goal is to help readers develop an intuitive understanding of the principles and techniques that enable these models to produce high-quality and diverse results.

Resumo

Modelos de difusão constituem o paradigma dominante para a síntese de imagens e vídeos de alta qualidade. Esses modelos geram conteúdos visuais ao reverter um processo estocástico gradual que transforma dados de treinamento em ruído Gaussiano, enquanto condicionam o processo de geração por meio de instruções, como o uso de descrições textuais. Este tutorial oferece uma introdução acessível porém tecnicamente rigorosa a modelos de difusão para síntese de imagens. Seu objetivo é apresentar de forma intuitiva os princípios e técnicas que lhes permitem produzir resultados de alta qualidade.

Este trabalho foi financiado com recursos do CNPq (Processo 305474/2022-7) e CAPES (Finance Code 001). Ferramentas de IA Generativa, incluindo ChatGPT e Gemini foram empregadas para geração de ilustrações. ChatGPT foi utilizado na revisão textual.

2.1. Introdução

Sistemas de IA generativa como Midjourney [Midjourney, Inc. 2026], Nano Banana Pro [Google DeepMind 2025], DALL·E 3 [Betker et al. 2023], Imagen [Saharia et al. 2022], Stable Diffusion [Rombach et al. 2022] e Flux [Black Forest 2026] são capazes de produzir imagens de alta qualidade em resposta a uma descrição textual apresentada pelo usuário. Por sua vez, sistemas como Sora [OpenAI 2026], Runway Gen-4.5 [Runway AI 2026], Kling [Kuaishou 2026], Luma Dream Machine [Luma Labs 2026], Deevid [Deevid AI 2026] e Wan [Alibaba 2026] geram vídeos de alta qualidade a partir de *prompts* de texto. Dentro de suas respectivas classes, cada um desses sistemas produz resultados que se destacam por certas características. Apesar de suas diferenças, todos eles têm algo em comum: utilizam **modelos de difusão** como mecanismo para síntese de imagens. Modelos de difusão constituem o paradigma dominante para a síntese de imagens e vídeos de alta qualidade a partir de descrições textuais. A Figura 2.1 mostra uma imagem gerada por um modelo de difusão com base em uma descrição. Observe a riqueza de detalhes gerada para esta cena virtual. O objetivo deste tutorial é apresentar de forma intuitiva, porém tecnicamente rigorosa, os princípios e as técnicas que tornam isso possível.

Figura 2.1. Imagem gerada utilizando um modelo de difusão (DALL·E 3 [Betker et al. 2023]) com a descrição: "Um ursinho andando de bicicleta em um parque, usando óculos escuros, um boné azul e uma camisa com a palavra INF".

2.2. Modelagem Generativa Como Um Processo de Amostragem

Como é possível gerar imagens como a mostrada na Figura 2.1 a partir de uma descrição textual? A resposta é: por meio de um **processo de amostragem** condicionada. Mas antes que possamos explicá-lo em detalhes, é necessário entender como gerar imagens (*i.e.*, como realizar amostragem) sem condicionamento ao texto. Vamos iniciar introduzindo algumas ideias-chave, a partir das quais o procedimento se torna bastante intuitivo.

Ideia-chave 1: **Modelagem generativa** pode ser entendida como o **processo de aprender uma distribuição de probabilidades** a partir de um conjunto de dados utilizado para treinamento, de modo que novas amostras semelhantes às deste conjunto de dados possam ser obtidas (geradas) amostrando-se esta distribuição. Uma **distribuição de probabilidades** para um conjunto de dados de treinamento é uma descrição matemática de quão prováveis são diferentes amostras de acordo com os dados utilizados no treinamento.

Modelos de difusão podem ser utilizados para geração de dados de diversas modalidades, como imagens [Betker et al. 2023], vídeos [OpenAI 2026], modelos 3D [Wang et al. 2025], nuvens de pontos [Luo and Hu 2021] e estruturas moleculares [Abramson et al. 2024], entre outras. Todas essas modalidades podem ser manipuladas utilizando representações vetoriais. Assim, uma imagem contendo H linhas, W colunas e C canais de cores pode ser representada por um vetor $z \in R^{H \times W \times C}$. Um vídeo corresponde a uma sequência de imagens ao longo de um período de tempo T e pode ser representado por um vetor $z \in R^{T \times H \times W \times C}$. Por sua vez, um modelo 3D, uma nuvem de pontos, ou uma estrutura molecular pode ser representado(a) por um vetor $z \in R^{3 \times N}$, onde N corresponde, respectivamente, ao número de vértices do modelo 3D, ao número de pontos da nuvem, ou ao número de átomos da molécula, cada um desses correspondendo a um elemento $w \in R^3$. No restante do texto, focaremos no caso de geração de imagens e, assim, o termo *amostra* será interpretado como *imagem*. Ressalte-se, entretanto, que os conceitos fundamentais sobre modelos de difusão e o processo aqui descrito para imagens se aplicam de forma similar às demais modalidades, diferindo apenas na representação dos dados, nas arquiteturas de redes neurais utilizadas para estimar ruído, e nos mecanismos de condicionamento, todos adaptados para cada domínio específico.

Ideia-chave 2: A obtenção da distribuição de probabilidades verdadeira (*true probability distribution*) p_{dados} para um conjunto de imagens não pode ser obtida a partir de número finito de amostras. Isto ocorre porque: (i) tipicamente, o processo gerador das amostras é desconhecido; (ii) o espaço no qual as amostras encontram-se representadas tem altíssima dimensionalidade (para imagens, $z \in R^{H \times W \times C}$); (iii) Uma quantidade muito pequena de amostras (relativamente às dimensões do espaço) está disponível e a maioria das amostras da distribuição verdadeira nunca será observada. Assim, **o processo generativo precisa "aprender" uma aproximação $\hat{p}_{\text{dados}}$ para p_{dados}** a partir das amostras de treinamento.

Um modelo de difusão aprende a distribuição $\hat{p}_{dados}$ iterando um procedimento que realiza adição e remoção de ruído às imagens de treinamento. Em um primeiro estágio, o **processo de difusão** (*forward diffusion process*) **adiciona ruído** Gaussiano com média zero e variância isotrópica unitária às imagens de treinamento x_0 até que estas se convertam em puro ruído x_T caracterizado por uma aproximação a uma distribuição normal padrão $\mathcal{N}(0, I)$. No segundo estágio, o processo de **difusão reversa aprende a estimar e remover o ruído presente nas imagens** ou, equivalentemente, aprende a mover amostras ruidosas nas direções do espaço mais prováveis de acordo com a distribuição $\hat{p}_{dados}$. O processo de **difusão reversa** constitui a **essência do modelo generativo**. Ao aplicar a remoção de ruído aprendida durante o treinamento a outras amostras ruidosas, o modelo reconstrói novas imagens que seguem a estrutura da distribuição $\hat{p}_{\text{dados}}$. O mecanismo que garante que o processo de remoção de ruído conduza amostras ruidosas x_T em direção a amostras $\hat{x}_0 \sim \hat{p}_{\text{dados}}$ é a existência de um campo vetorial chamado **função escore** que será discutido na Seção 2.3.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

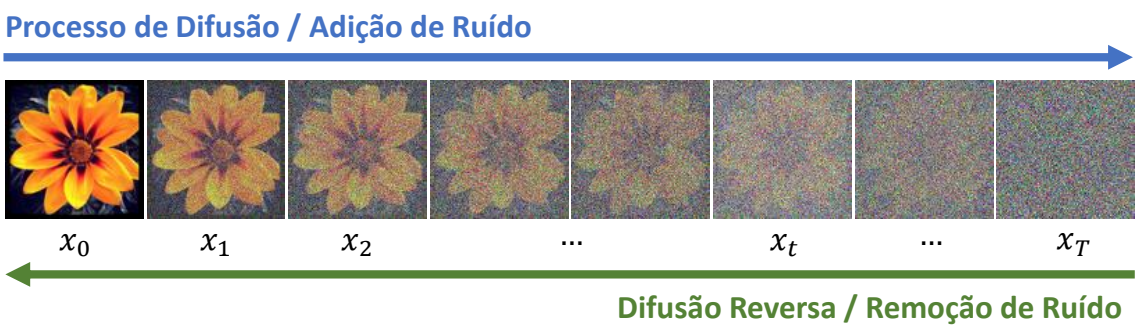

Figura 2.2. Processo de difusão: gradual atenuação do conteúdo e adição de ruído Gaussiano $\varepsilon \sim \mathcal{N}(0,I)$ à imagem de uma flor. Após T passos, a imagem x_0 é transformada em puro ruído $x_T \approx \mathcal{N}(0,I)$. O processo de difusão reversa remove gradualmente o ruído, restaurando a imagem original.

Os processos de adição e de remoção de ruído são ilustrados na Figura 2.2. Nela, durante o processo de difusão a imagem de uma flor (x_0) é progressivamente corrompida pela gradual atenuação de seu conteúdo e adição de ruído Gaussiano $\varepsilon \sim \mathcal{N}(0,I)$ até que, ao final de T passos, a representação resultante tenha se convertido em puro ruído $x_T \approx \mathcal{N}(0,I)$. O processo de difusão reversa remove gradualmente o ruído.

Distribuições Gaussianas estão no centro de modelos de difusão visto que possuem diversas propriedades que garantem que estes sejam matematicamente tratáveis. Elas são completamente caracterizadas utilizando apenas média e covariância, possuem representação analítica, sua amostragem é simples, e possuem função escore definida analiticamente. Outra aspecto fundamental: elas tornam o processo de adição de ruído simples, permitindo a geração de amostras com níveis arbitrários de ruído diretamente a partir de amostras de treinamento. O processo reverso pode ser aprendido de forma gradual e estável. Outro aspecto relevante é o fato do ruído Gaussiano ser isotrópico e suave, não introduzindo viés direcional no espaço. Assim, a distribuição normal padrão $\mathcal{N}(0,I)$ **é utilizada como ponto de partida** x_T para o processo de difusão reversa. Os termos difusão reversa e processo de amostragem são usados como sinônimos ao longo do texto.

Para avançarmos na compreensão do funcionamento de um modelo de difusão para síntese de imagens, vamos agora aplicar o processo de adição gradual de ruído Gaussiano às amostras de um conjunto de treinamento. Seja $\mathscr{D} = \{x_0^{(i)}\}_{i=1}^{N}$ o conjunto de imagens de treinamento, onde N representa o número total de imagens e $x_0^{(i)}$ i-ésima imagem. O processo de adição de ruído é ilustrado na Figura 2.3. À esquerda, há três agrupamentos de pontos azuis, onde cada ponto representa uma imagem (um vetor $x_0^{(i)} \in R^{H\times W\times C}$). Cada agrupamento contém imagens de apenas uma entre três classes distintas: flores, gatos e cães. Coletivamente, essas amostras constituem uma distribuição multimodal. Lembre-se que o objetivo do processo generativo é ser capaz de produzir novas amostras similares às do conjunto de treinamento. Assim, o modelo precisará aprender uma aproximação empírica $\hat{p}_{\text{dados}}$ para p_{dados}, estimada diretamente a partir das amostras observadas.

O processo de difusão aplica uma gradual atenuação do conteúdo original das imagens e adição de ruído Gaussiano $\varepsilon \sim \mathcal{N}(0,I)$ a cada uma das amostras do conjunto de treinamento $\mathscr{D}$ ao longo de T passos. A evolução dessas imagens é representada por

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

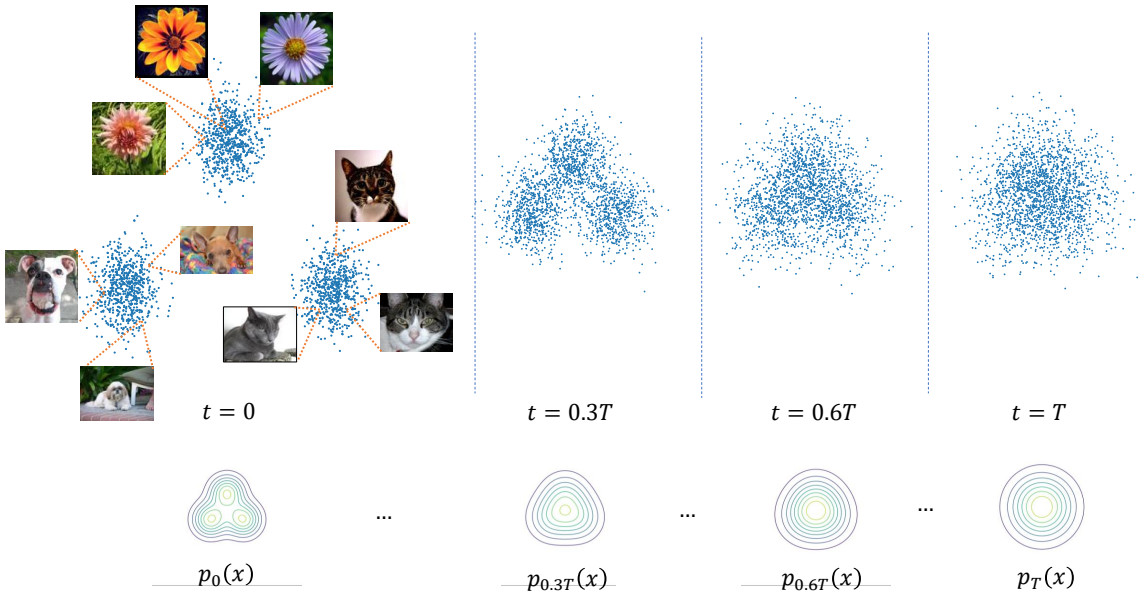

Figura 2.3. Representação abstrata do processo de difusão aplicado a imagens do conjunto de treinamento. As amostras de uma distribuição arbitrária contendo imagens de flores, gatos e cachorros (em $t = 0$) são convertidas em em amostras de uma distribuição Gaussiana $p_T \approx \mathcal{N}(0, I)$ em $t = T$. Isto ocorre por meio da gradual atenuação do conteúdo original das imagens e adição de ruído Gaussiano com média zero e variância isotrópica unitária ao longo de T passos, para T suficientemente grande. Na parte inferior, curvas de nível representam as funções de densidade de probabilidade $p_t(x)$ para diversos valores de t.

$x_t^{(i)}$, com $t \in \{0, 1, 2, ..., T\}$, onde $x_0^{(i)}$ corresponde a uma amostra do conjunto de treinamento (*i.e.*, sem ruído) e $x_T^{(i)}$ corresponde à imagem obtida a partir de $x_0^{(i)}$ após os T passos. Considerando T suficientemente grande, este processo transforma cada imagem $x_0^{(i)}$ em uma imagem ruidosa $x_T^{(i)} \approx \mathcal{N}(0, I)$. Ou seja, o processo de difusão transforma gradualmente amostras da distribuição desconhecida p_{dados} em amostras de uma distribuição que aproxima a distribuição normal padrão $\mathcal{N}(0, I)$. Esta situação é ilustrada na Figura 2.3, onde as amostras da distribuição multimodal $p_0 = p_{dados}$ à esquerda são gradualmente transformadas em amostras de uma distribuição Gaussiana $p_T \approx \mathcal{N}(0, I)$, à direita. Representações também são mostradas para $t = 0,3T$ e $t = 0,6T$.

2.3. A Função Escore

Conforme mencionado, a essência do processo generativo corresponde a mover amostras ruidosas nas direções do espaço mais prováveis de acordo com a distribuição empírica $\hat{p}_{dados}$. Mas como determinar essas direções?

Ideia-chave 3: O log do gradiente da função de densidade de probabilidade em cada uma das etapas do processo no qual t varia de 0 a T fornece um campo vetorial $s_\theta(x_t, t)$ que permite guiar amostras ruidosas em $p_T \approx \mathcal{N}(0, I)$ de volta à $\hat{p}_{\text{data}}$.

A porção inferior da Figura 2.3 mostra as funções de densidade $p_t(x)$ associadas às distribuições de probabilidade p_t para os diversos valores de t ao longo do processo de difusão. Na figura, cada função de densidade de probabilidade é representada por

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

um conjunto de curvas de nível, nas quais os contornos mais internos correspondem a densidades maiores. Assim, o gradiente $\nabla_x p_t(x)$ dessas funções de densidade fornece, para cada distribuição marginal[1] p_t (*i.e.*, para cada estágio do processo de difusão), um campo vetorial que aponta para as regiões de maior concentração de amostras no estágio t. Tomados coletivamente de $t = T$ até $t = 0$, esses campos vetoriais definiriam o caminho para guiar as amostras $p_T(x) \approx \mathcal{N}(0, I)$ da distribuição Gaussiana terminal de volta para as regiões de maior densidade, e portanto mais prováveis, da distribuição $\hat{p}_{\text{dados}}$.

Entretanto, em espaços de alta dimensionalidade os valores de $p_t(x)$ tendem a ser extremamente pequenos em quase todo o espaço. Com isso, $\nabla_x p_t(x)$ resulta em valores muitíssimo pequenos, especialmente em regiões onde a densidade de dados é baixa. Nessas condições, as magnitudes desses campos vetoriais tendem a zero, fazendo com que o sinal disponível para guiar o processo reverso seja muito fraco, tornando o processo generativo (amostragem de $\hat{p}_{\text{data}}$) instável e inefetivo. Este problema é resolvido normalizando o gradiente pelas respectivas densidades:

$$\nabla_x \log p(x) = \frac{\nabla_x p(x)}{p(x)}. \tag{1}$$

A Equação 1 define a **função escore** (*score function*) **responsável por guiar as amostras** x_T **para** $\hat{x}_0$, que caracteriza o aspecto generativo dos modelos de difusão. Ao normalizar os gradientes pelas respectivas densidades, esta função preserva a informação direcional do campo mesmo onde as probabilidades assumem valores muito pequenos. A Figura 2.4 ilustra o uso da função escore para obtenção de um campo vetorial. A imagem à esquerda mostra uma função de densidade de probabilidade $p(x)$ representada por curvas de nível, onde os tons de vermelho e laranja correspondem a regiões de maior densidade. A imagem à direita exibe o campo vetorial definido pela função escore para $p(x)$. Localmente, o campo aponta em direção às maiores densidades.

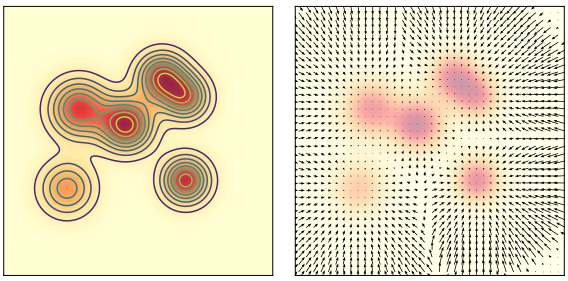

Figura 2.4. Uma função de densidade de probabilidade $p(x)$ representada utilizando curvas de nível. Os tons de vermelho e laranja correspondem às regiões de maior densidade (esquerda). Campo vetorial obtido aplicando-se a função escore a $p(x)$. Localmente, os vetores apontam em direção às maiores densidades.

2.3.1. Gerando Imagens a Partir da Função Escore

Utilizando a função escore, o processo de síntese de imagens se torna intuitivo, bastando para isso gerar uma amostra ruidosa $x_T \sim p_T$ e aplicar a ela o processo de difusão reversa. Como $p_T \approx \mathcal{N}(0, I)$, obtemos x_T amostrando diretamente a distribuição normal padrão:

[1] Cada distribuição individual p_t é chamada de uma *distribuição marginal*.

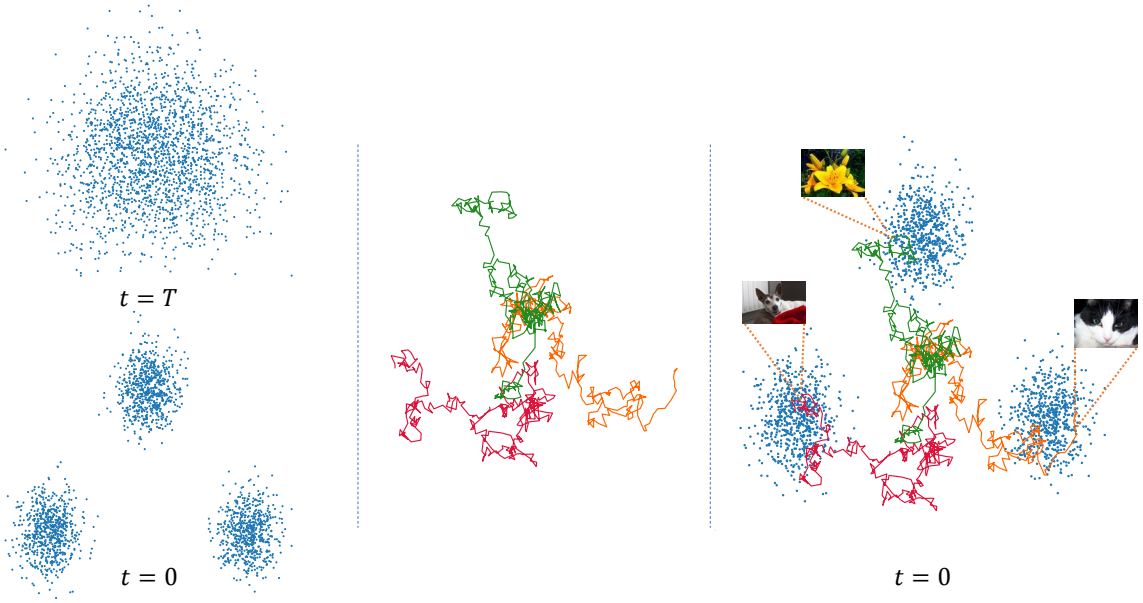

Figura 2.5. Processo de difusão reversa utilizado pela técnica DDPM. (esquerda) Nuvens de pontos representando a distribuição de imagens $p_0 = \hat{p}_{dados}$ no tempos $t = 0$ e e p_T no tempo $t = T$. (centro) trajetórias para três amostras x_t partindo de $x_T \sim \mathcal{N}(0, I)$ até $\hat{x}_0$. (direita) Trajetórias superpostas à nuvem representando $\hat{p}_{dados}$ e as imagens reconstruídas no processo.

$x_T \sim \mathcal{N}(0, I)$. Este processo é ilustrado na Figura 2.5. À esquerda, tem-se as nuvens de pontos que representam as amostras utilizadas no exemplo da Figura 2.3 posicionadas nos instantes $t = 0$ (abaixo) e $t = T$ (acima). Ao centro, vemos as trajetórias de três amostras x_t partindo de $x_T \sim \mathcal{N}(0, I)$ até $\hat{x}_0$. À direita, tem-se as trajetórias superpostas às nuvens de pontos representando $\hat{p}_{dados}$ e as respectivas imagens reconstruídas no processo.

2.3.2. Modelagem Generativa Baseada em Modelos de Difusão

As ideias fundamentais associadas a modelagem generativa baseada em difusão envolvendo corromper progressivamente os dados de treinamento pela adição de ruído Gaussiano e aprender um processo estocástico reverso para geração de novas amostras, descritas na Seção 2.2, foram introduzidas por Sohl-Dickstein e colaboradores [Sohl-Dickstein et al. 2015]. No artigo, os autores não utilizaram a função escore para guiar o processo reverso, tendo utilizado uma rede neural para reverter cada passo do processo de difusão.

Ho e colaboradores [Ho et al. 2020] introduziram diversas melhorias à técnica descrita em [Sohl-Dickstein et al. 2015], tornando modelos de difusão mais fáceis de treinar e melhorando significativamente a qualidade das imagens geradas. A técnica de Ho também não utiliza explicitamente a função escore. Ela treina uma rede neural para predizer o ruído Gaussiano ε acrescido a cada amostra x_t. Uma interpretação explícita do processo de difusão reversa baseado na função score e sua conexão com modelos de difusão tornou-se clara através dos trabalhos de Song e colaboradores [Song and Ermon 2019; Song et al. 2020]. A técnica introduza por Ho e colaboradores, conhecida como **DDPM** (acrônimo para *Denoising Diffusion Probabilistic Models*) inaugurou o que se convencionou chamar de sistemas de difusão modernos.

2.4. Abordagem Probabilística para Modelos de Difusão

Esta seção detalha a técnica DDPM, descrevendo o algoritmo utilizado para treinamento da rede neural utilizada para estimar o ruído em cada amostra x_t, bem como o algoritmo que implementa o processo de difusão reversa propriamente dito.

Seja $x_0 \to x_1 \to \cdots \to x_T$ uma sequência de amostras obtida durante a etapa de adição gradual de ruído Gaussiano. Este processo é definido pela distribuição Gaussiana condicional $q(x_t \mid x_{t-1}) = \mathcal{N}\left(x_t; \sqrt{1-\beta_t}\,x_{t-1}, \beta_t I\right)$. Neste caso, x_{t-1} e x_t pertencem à mesma trajetória de difusão, sendo x_t obtido a partir de x_{t-1} por meio da expressão

$$x_t = \sqrt{1-\beta_t}\,x_{t-1} + \sqrt{\beta_t}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I), \tag{2}$$

onde β_t é a variância do ruído Gaussiano acrescentado na etapa t, tal que $\beta_t \in (0,1)$ e $\beta_1 < \beta_2 < \cdots < \beta_T$. A sequência de valores de β_t, chamada de *schedule* de variância (ou *schedule* de ruído), é um parâmetro da técnica DDPM. De acordo com a Equação 2, o valor da amostra x_t é obtido atenuando-se o valor de x_{t-1} e acrescentando-se ruído Gaussiano. Assim, à medida que o valor de β_t se aproxima de 1, x_t se aproxima de ruído Gaussiano com média zero e variância unitária. Por pertencerem à mesma trajetória de difusão, poderíamos escrever $x_t^{(i)} = \sqrt{1-\beta_t}\,x_{t-1}^{(i)} + \sqrt{\beta_t}\,\varepsilon$ para explicitar que se trata de estágios consecutivos da transformação da amostra $x_0^{(i)}$ do conjunto de treinamento.

Fazendo $\alpha_t = 1 - \beta_t$, a Equação 2 pode ser reescrita como

$$x_t = \sqrt{\alpha_t}\,x_{t-1} + \sqrt{1-\alpha_t}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I). \tag{3}$$

Definindo $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$ e $\bar{\beta}_t = 1 - \bar{\alpha}_t$, pode-se expressar o valor de x_t diretamente a partir de x_0, conforme a Equação 4:

$$x_t = \sqrt{\bar{\alpha}_t}\,x_0 + \sqrt{1-\bar{\alpha}_t}\,\varepsilon, \qquad \varepsilon \sim \mathcal{N}(0, I). \tag{4}$$

A possibilidade de obter amostras x_t com níveis arbitrários de ruído diretamente a partir das imagens x_0 simplifica e acelera o processo de treinamento da rede neural ε_θ utilizada pela técnica DDPM para estimar a quantidade de ruído presente em x_t. Este treinamento é descrito no Algoritmo 1. Ele itera sobre imagens x_0 do conjunto de treinamento, e amostra uniformemente um nível de ruído t. Uma amostra ruidosa x_t é então obtida de acordo com a Equação 4 e passada juntamente com o valor de t para a rede neural que estima $\varepsilon_\theta(x_t, t)$, o ruído presente em x_t. Observe que o treinamento de ε_θ é realizado gerando-se amostras x_t para valores de t definidos estocasticamente, não requerendo percorrer os valores de t na sequência de 0 a T, como representado na etapa de difusão ilustrada na Figura 2.2.

A função de perda L utilizada pelo Algoritmo 1 calcula o erro quadrático médio entre a quantidade de ruído ε adicionada a x_t e seu valor estimado $\varepsilon_\theta(x_t, t)$ para as várias imagens em um lote. Observem que tanto ε quanto $\varepsilon_\theta(x_t, t)$ são tensores definidos no espaço $R^{H \times W \times C}$, ao passo que L é um escalar. Assim, o valor de L utilizado para atualização dos pesos da rede neural ε_θ é obtido em dois passos: para cada amostra ruidosa x_t em um lote (*minibatch*), calcula-se o erro quadrático médio (*MSE*) entre os tensores ε e $\varepsilon_\theta(x_t, t)$. O valor de L é obtido como a média destes erros para todas as amostras do lote. O mecanismo de retropropagação (*backpropagation*) então computa os gradientes da função L com relação aos parâmetros θ da rede neural e um otimizador (*e.g.*, SGD

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

ou Adam) realiza um passo de atualização dos pesos da rede utilizando o algoritmo de descida do gradiente (linha 8 do Algoritmo 1). Este processo é repetido para diversas amostras x_0 e valores de t até que o valor de L e a qualidade das imagens geradas (considerando aspectos como qualidade, coerência e diversidade) se estabilizem. Observem que o Algoritmo 1 apenas otimiza a predição de ruído e não gera imagens diretamente. Para avaliar a qualidade visual das imagens geradas durante a fase de treinamento, os pesos do modelo são salvos periodicamente e utilizados para executar o procedimento de amostragem (Algoritmo 2) que gera imagens a partir de $x_T \sim \mathcal{N}(0,I)$: $x_T \to x_{T-1} \to \cdots \to \hat{x}_0$. As imagens geradas são então inspecionadas visualmente ou avaliadas utilizando métricas como FID (*Fréchet inception distance*) [Heusel et al. 2017]. FID é computada entre as imagens geradas $\hat{x}_0$ e imagens x_0 do conjunto de treinamento.

Algoritmo 1: Treinamento da rede neural ε_θ utilizada por DDPM estimar ruído em x_t.

1: **repeat**
2: $x_0 \sim p_{\text{data}}(x)$ // Amostra uma imagem do conjunto de treinamento
3: $t \sim \text{Uniform}(\{1,\ldots,T\})$ // Amostra um valor de passo t uniformemente
4: $\varepsilon \sim \mathcal{N}(0,I)$ // Gera ruído Gaussiano
5: $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\varepsilon$ // Gera amostra ruidosa a partir de x_0
6: Calcula função de perda e executa um passo de descida do gradiente
7: $L = \|\varepsilon - \varepsilon_\theta(x_t,t)\|^2$ // Avalia a função de perda (loss function)
8: $\theta \leftarrow \theta - \eta\nabla_\theta L$ // Atualiza os pesos da rede ε_θ utilizando descida do gradiente
9: **until** convergência

Algoritmo 2: Procedimento de amostragem utilizado pela técnica DDPM.

1: $x_T \sim \mathcal{N}(0,I)$ // Inicializa $x_T \sim \mathcal{N}(0,I)$ a partir de ruído Gaussiano
2: **for** $t = T,\ldots,1$ **do** // Percorre sequência reversa com t variando de T até 1
3: **If** $t > 1$ **then** $z \sim \mathcal{N}(0,I)$ **else** $z = 0$ // Ruído estocástico a ser reinjetado em x_{t-1}
4: $\hat{\varepsilon} = \varepsilon_\theta(x_t,t)$ // Estima o ruído presente em x_t usando a rede neural ε_θ
5: $\mu_\theta(x_t,t) = \frac{1}{\sqrt{\alpha_t}}\left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\hat{\varepsilon}\right)$ // Estimativa mais provável de x_{t-1} que gerou x_t
6: $x_{t-1} = \mu_\theta(x_t,t) + \sigma_t z$ // Estima x_{t-1} com $\sigma_t < \sigma_{t+1}$
7: **end for**
8: **return** x_0 // Retorna a imagem gerada

O Algoritmo 2 detalha o processo de amostragem utilizado pela técnica DDPM e que efetivamente realiza a modelagem generativa. Partindo-se de uma amostra x_T obtida a partir de uma distribuição normal padrão (linha 1), o processo de remoção de ruído é aplicado à toda a sequência $x_T \to x_{T-1} \to \cdots \to \hat{x}_0$ (processo reverso mostrado na Figura 2.2). A rede neural treinada pelo Algoritmo 1 é utilizada para estimar a quantidade total de ruído presente na amostra x_t: $\varepsilon_\theta(x_t,t)$ (linha 4). Este ruído então é utilizado para obter $\mu_\theta(x_t,t)$ (linha 5), que corresponde à estimativa mais provável para o valor de x_{t-1} que teria produzido a amostra x_t após um passo de adição de ruído. O novo valor de x_{t-1} é então atualizado pela injeção de ruído Gaussiano com desvio padrão σ_t (linha 6), onde $\sigma_t < \sigma_{t+1}$. Esta adição de ruído permite a geração de resultados mais diversificados, ao invés de convergir deterministicamente de um dado valor de x_T para uma imagem $\hat{x}_0$.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

2.4.1. A Rede Neural Utilizada por DDPM Aproxima a Função Escore

A obtenção de x_{t-1} a partir de x_t por meio de $\mu_\theta(x_t,t)$ segue o campo vetorial definido pela função escore. Podemos verificar este fato observando que dada uma função Gaussiana multivariada normalizada com média μ e variância σ^2

$$p(x) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^T(x-\mu)\right),$$

sua função escore corresponde a

$$\nabla_x \log p(x) = -\frac{1}{\sigma^2}(x-\mu).$$

De acordo com a Equação 4, a etapa de difusão implementada pela técnica de DDPM define a distribuição condicional

$$p(x_t \mid x_0) = \mathcal{N}\left(x_t; \sqrt{\bar{\alpha}_t}\, x_0, (1-\bar{\alpha}_t)I\right),$$

para a qual a função escore com relação a x_t é dada por

$$\nabla_{x_t} \log p(x_t \mid x_0) = -\frac{x_t - \sqrt{\bar{\alpha}_t}\, x_0}{1-\bar{\alpha}_t}. \tag{5}$$

Novamente, de acordo com a Equação 4, o valor de x_0 pode ser aproximado utilizando o ruído estimado por $\varepsilon_\theta(x_t,t)$ como

$$x_0 \approx \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(x_t,t)}{\sqrt{\bar{\alpha}_t}}. \tag{6}$$

Substituindo a Equação 6 na Equação 5, obtem-se:

$$\nabla_{x_t} \log p(x_t \mid x_0) \approx -\frac{x_t - \left(x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(x_t,t)\right)}{1-\bar{\alpha}_t} = -\frac{\sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(x_t,t)}{1-\bar{\alpha}_t} = -\frac{\varepsilon_\theta(x_t,t)}{\sqrt{1-\bar{\alpha}_t}}. \tag{7}$$

A Equação 7 mostra que a **função escore** com relação a x_t para a distribuição condicional $p(x_t \mid x_0)$ utilizada no processo de difusão da técnica DDPM é **proporcional ao ruído $\varepsilon_\theta(x_t,t)$ estimado pela rede neural**. Este fato garante a convergência e a estabilidade da técnica de DDPM. A linha 5 do Algoritmo 2 obtém $\mu_\theta(x_t,t)$, a estimativa mais provável para x_{t-1}, somando a x_t uma versão escalada da função escore e dividindo o resultado por $\sqrt{\alpha_t}$. Esta divisão compensa o fato de que x_t é obtido multiplicando x_{t-1} por $\sqrt{\alpha_t}$ durante a etapa de adição de ruído (Equação 3).

As arquiteturas de redes neurais mais comumente utilizadas na implementação modelos de difusão são baseadas na arquitetura U-Net [Ronneberger et al. 2015] e mais recentemente em Diffusion Transformers (DiT) [Peebles and Xie 2023]. Para o processo de amostragem descrito no Algoritmo 2, é necessário percorrer toda a sequência de T passos $x_T \to x_{T-1} \to \cdots \to \hat{x}_0$. A técnica clássica de DDPM utiliza $T = 1000$ como valor padrão para geração de imagens de alta qualidade. Isto torna o processo de amostragem relativamente lento, visto que a rede neural precisa ser avaliada a cada passo. A Sessão 2.5 e a discussão sobre *difusão latente* na Seção 2.6.3 apresentam estratégias para acelerar o processo de amostragem. Naturalmente, fatores como resolução das imagens, tamanho do modelo e hardware utilizado também afetam o tempo de processamento.

2.5. Abordagem Implícita para Modelos de Difusão

Song e colaboradores [Song et al. 2021] observaram que um modelo de difusão treinado ao estilo DDPM pode ser amostrado utilizando um **processo reverso determinístico** produzindo imagens de alta qualidade. Com base nesta observação, propuseram a técnica **DDIM** (acrônimo para *Denoising Diffusion Implicit Models*) que reformula a etapa de difusão reversa, reduzindo significativamente o número de passos necessários para gerar uma imagem. A **ideia chave da técnica DDIM** é estimar diretamente o valor de $\hat{x}_0$ a partir de qualquer amostra x_t, isolando x_0 na Equação 4:

$$\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\varepsilon_\theta(x_t,t)}{\sqrt{\bar{\alpha}_t}}, \tag{8}$$

e utilizá-lo para gerar novas amostras ruidosas para qualquer passo $t-k$:

$$x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-k}}\,\varepsilon_\theta(x_t,t). \tag{9}$$

A Equação 9 pode ser interpretada geometricamente como definindo a posição da nova amostra ruidosa x_{t-k} no espaço das transformações. Lembre-se que $\varepsilon_\theta(x_t,t)$ aproxima o campo vetorial reverso definido pela função escore para a distribuição marginal p_t em x_t, e que em DDPM/DDIM α_t, β_t e $\bar{\alpha}_t$ são valores escalares. Assim, x_{t-k} é obtida "movendo-se" uma versão escalada de $\hat{x}_0$ na direção da amostra x_t.

O processo de treinamento da rede neural ε_θ é o mesmo utilizado por DDPM e descrito no Algoritmo 1. Já o processo de amostragem realizado por DDIM consiste em inicializar $x_T \sim \mathcal{N}(0,I)$ e iterar as Equações 8 e 9 utilizando passos de tamanho k. Neste caso, DDIM percorre a sequência $x_T \to x_{T-k} \to x_{T-2k} \to \cdots \to \hat{x}_0$, utilizando um número consideravelmente menor de passos para obter uma amostra $\hat{x}_0$. O uso de 50 passos de difusão reversa (*e.g.*, $T = 1000$ e $k = 20$) tende a produzir resultados de ótima qualidade. Entretanto, amostradores modernos tendem a utilizar uma abordagem adaptativa, com passos menores ao final do processo, possibilitando um melhor refinamento dos detalhes das imagens. O processo de amostragem descrito é determinístico, significando que para um mesmo valor de x_T e mesma rede ε_θ, DDIM produzirá sempre a mesma imagem $\hat{x}_0$. Esta situação pode ser contornada acrescentando-se um termo estocástico à Equação 9:

$$x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-k}}\,\varepsilon_\theta(x_t,t) + \sigma_t z, \quad z \sim \mathcal{N}(0,I), \tag{10}$$

onde σ_t representa o desvio padrão do ruído Gaussiano acrescentado. Quando $\sigma_t = 0$, tem-se o caso determinístico; valores de $\sigma_t > 0$ introduzem randomicidade ao processo de amostragem. O Algoritmo 3 descreve o processo de amostragem utilizado por DDIM.

Algoritmo 3: Procedimento de amostragem utilizado pela técnica DDIM.

Require: σ_t

1: $x_T \sim \mathcal{N}(0,I)$ // Inicializa $x_T \sim \mathcal{N}(0,I)$ a partir de ruído Gaussiano
2: **for** $t = T,\ldots,1$ **step** k **do**
3: $\quad\hat{\varepsilon} = \varepsilon_\theta(x_t,t)$ // Estima o ruído presente em x_t usando a rede neural ε_θ
4: $\quad\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}}$ // Obtém a estimativa da imagem limpa $\hat{x}_0$
5: $\quad z \sim \mathcal{N}(0,I)$ // Ruído Gaussiano para versão estocástica, caso $\sigma_t > 0$
6: $\quad x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-k}}\,\hat{\varepsilon} + \sigma_t z$ // Executa etapa de difusão reversa
7: **end for**
8: **return** $\hat{x}_0$ // Retorna a imagem gerada

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

2.6. Condicionando Modelos de Difusão

Os procedimentos descritos até aqui geram imagens considerando apenas a distribuição de dados aprendida $\hat{p}_{\text{dados}}$, sem um mecanismo que permita direcioná-las para geração de conteúdo específico. O exemplo da Figura 2.5 (direita) ilustra esta situação. Nele, os três valores sementes x_T geraram uma imagem de flor, uma de gato e outra de cachorro. Entretanto, essas imagens foram geradas de modo automático sem escolha do usuário.

É possível direcionar os resultados produzidos por modelos de difusão utilizando instruções para gerar conteúdo com características desejadas. Essas instruções ou comandos, chamados de *prompts*, podem incluir textos, imagens, áudios, ou uma combinação destes (*prompts* multimodais). A Figura 2.1 ilustra a geração de uma imagem utilizando o comando textual "*Um ursinho andando de bicicleta em um parque, usando óculos escuros, um boné azul e uma camisa com a palavra INF*". Note que, a menos que o modelo de difusão utilize um processo de amostragem determinístico, seja inicializado com o mesmo valor inicial de x_T e utilize os mesmos parâmetros da rede neural ε_θ, um mesmo *prompt* gerará imagens diferentes a cada execução.

2.6.1. Gerando Imagens Guiadas por Descrições Textuais

O processo de geração de imagens guiadas por descrições é conceitualmente bastante semelhante ao apresentado nas seções anteriores. Essencialmente, envolve apenas a substituição da função de distribuição de probabilidades $p(x)$ por **uma função de distribuição de probabilidade condicional** $p(x|c)$, onde c é a informação condicionante. No caso de geração de imagens a partir de descrições textuais, c corresponde a um *prompt* textual. Os processos de difusão e de difusão reversa são, consequentemente, semelhantes aos definidos para DDPM e DDIM nas Seções 2.4 e 2.5.

Assim como nos casos clássicos envolvendo DDPM e DDIM, a obtenção de um modelo de difusão condicional consiste em treinar uma rede neural para estimar o ruído $\varepsilon_\theta(x_t, t, c)$ adicionado à amostra ruidosa x_t. A etapa de difusão reversa segue então os campos vetoriais definidos pelas funções escore condicionais $s_\theta(x_t, t, c) \approx \nabla_{x_t} \log p_t(x_t \mid c)$ aprendidas por ε_θ, ao invés de $s_\theta(x_t, t) \approx \nabla_{x_t} \log p_t(x_t)$. Os procedimentos para treinamento e amostragem do modelo estão descritos nos Algoritmos 4 e 5, respectivamente.

Algoritmo 4: Treinamento da rede neural $\varepsilon_\theta(x_t, t, c)$ que estima ruído em x_t para um Modelo de Difusão Condicionado por Texto.

repeat
  $(x_0, c) \sim p_{\text{data}}(x, c)$ // Amostra uma imagem de treinamento x_0 e sua descrição c
  $t \sim \text{Uniform}(\{1, \ldots, T\})$ // Amostra um valor de passo t uniformemente
  $\varepsilon \sim \mathcal{N}(0, I)$ // Gera ruído Gaussiano
  $x_t = \sqrt{\bar{\alpha}_t}\, x_0 + \sqrt{1 - \bar{\alpha}_t}\, \varepsilon$ // Gera amostra ruidosa a partir de x_0
  Calcula função de perda e executa um passo de descida do gradiente
  $L = \|\varepsilon - \varepsilon_\theta(x_t, t, c)\|^2$ //Avalia a função de perda (loss function)
  $\theta \leftarrow \theta - \eta \nabla_\theta L$ // Atualiza os pesos da rede ε_θ utilizando descida do gradiente
until convergência

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Algoritmo 5: Amostragem no Estilo DDIM Condicionada por Texto.
Require: Texto c contendo descrição para a imagem e σ_t
1: $x_T \sim \mathcal{N}(0,I)$ // Inicializa $x_T \sim \mathcal{N}(0,I)$ a partir de ruído Gaussiano
2: **for** $t = T,\ldots,1$ **step** k **do**
3: $\hat{\varepsilon} = \varepsilon_\theta(x_t,t,c)$ // Estima o ruído presente em x_t usando a rede neural ε_θ
4: $\hat{x}_0 = \frac{x_t - \sqrt{1-\bar{\alpha}_t}\,\hat{\varepsilon}}{\sqrt{\bar{\alpha}_t}}$ // Obtém a estimativa da imagem limpa $\hat{x}_0$
5: $z \sim \mathcal{N}(0,I)$ // Ruído Gaussiano para versão estocástica, caso $\sigma_t > 0$
6: $x_{t-k} = \sqrt{\bar{\alpha}_{t-k}}\,\hat{x}_0 + \sqrt{1-\bar{\alpha}_{t-k}}\,\hat{\varepsilon} + \sigma_t z$ // Executa etapa de difusão reversa
7: **end for**
8: **return** $\hat{x}_0$

Lembre-se que a função escore define um campo vetorial para cada uma das T etapas do processo de difusão ou, mais precisamente, para cada uma das T distribuições marginais $p_t(x)$. Assim, é natural nos perguntarmos como c influencia o comportamento da rede neural ε_θ e, consequentemente, dos campos vetoriais $s_\theta(x_t,t,c)$. Para que isso ocorra, o *prompt* textual é convertido por um codificador de texto baseado em transformer [Vaswani et al. 2017] em representações numéricas (*embeddings*) que preservem a semântica do *prompt*. Essas representações são então usadas por camadas de atenção cruzada em ε_θ para guiar a geração de imagens.

O processo de condicionamento via *prompts* consiste em utilizar os *embeddings* do *prompt* para **modificar os mapas de ativação da rede** responsável por estimar o ruído em x_t. De modo sucinto, seja, por exemplo, $F(x_t,t)$ um conjunto de ativações (*features*) extraído por uma camada de uma rede U-Net a partir de uma imagem x_t. Um bloco transformer [Vaswani et al. 2017] na U-Net utiliza um mecanismo de atenção cruzada para obter $Q_F = F(x_t,t)W_Q$, $K_e = eW_K$, $V_e = eW_V$, onde e é o *embedding* do *prompt* c, e W_Q, W_K e W_V são as matrizes de projeção aprendidas que transformam vetores de características nas representações de *query*, *key*, e *value* usadas pelo mecanismo de atenção. Este obtém $A(x_t,t) = \text{softmax}\left(\frac{Q_F K_e^T}{\sqrt{d}}\right)V_e$. O valor A é então combinado a F por meio de uma conexão residual [He et al. 2016]: $F'(x_t,t) = F(x_t,t) + A(x_t,t)$ e $F'(x_t,t)$ é então passado para a camada subsequente da U-Net. Um bloco transformer é tipicamente inserido em múltiplas resoluções, incluindo vários níveis do decodificador, gargalo e codificador da U-Net. Isto permite que os *embeddings* de texto influenciem o processo de estimação de ruído e, portanto, as funções escore, ao longo de toda a rede.

A Figura 2.6 mostra um exemplo de imagem gerada por um modelo de difusão utilizando a descrição "*Um gato sentado em uma cadeira*". Note que a descrição menciona apenas um gato e uma cadeira, ao passo que a imagem reproduz um cenário bastante rico em detalhes. O mecanismo de atenção auxilia o modelo a organizar informações entre diferentes partes da imagem e entre a imagem e a descrição textual. Isto contribui para melhorar a coerência e composição da cena, além dos relacionamentos entre seus elementos. Entretanto, a imagem final resulta de propriedades que emergem da combinação de vários fatores, incluindo o treinamento do modelo de difusão, a estrutura estatística dos dados de treinamento, e o próprio mecanismo de atenção, entre outros.

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Figura 2.6. Imagem gerada utilizando o sistema Nano Banana Pro [?], utilizando a descrição: "Um gato sentado em uma cadeira".

2.6.2. Classifier-Free Guidance: Melhorando a Aderência das Imagens ao Prompt

Classifier-Free Guidance (CFG) é uma técnica utilizada durante o processo de difusão reversa para aumentar a influência do sinal de condicionamento c. Durante a etapa de treinamento, a rede ε_θ é ocasionalmente (*e.g.* com probabilidade entre 10% e 20%) treinada substituindo-se c por um *prompt* vazio ($c_\emptyset \leftarrow \emptyset$), enquanto a função de perda $L = \|\varepsilon - \varepsilon_\theta(x_t, t, c)\|^2$ permanece inalterada. Ao final do treinamento, a rede ε_θ terá aprendido a estimar tanto $\varepsilon_\theta(x_t, t, c)$ quanto $\varepsilon_\theta(x_t, t, \emptyset)$. Durante o processo de amostragem, o modelo avalia $\varepsilon_\theta(x_t, t, c)$ e $\varepsilon_\theta(x_t, t, \emptyset)$ e computa

$$\hat{\varepsilon} = \varepsilon_\theta(x_t, t, \emptyset) + w(\varepsilon_\theta(x_t, t, c) - \varepsilon_\theta(x_t, t, \emptyset)), \tag{11}$$

com $w \geq 1$ escalando o vetor guia $g = (\varepsilon_\theta(x_t, t, c) - \varepsilon_\theta(x_t, t, \emptyset))$. Ao amplificar a diferença entre as predições de ruído condicionada e não condicionada, CFG melhora a aderência das imagens geradas ao conteúdo do *prompt*.

2.6.3. Sistemas de Difusão Condicionados por Texto

Diversos sistemas foram desenvolvidos para geração de imagens a partir de descrições textuais. **Imagen** [Saharia et al. 2022], desenvolvido pela Google Research, produz imagens com alto grau de realismo, e elevado grau de compreensão e aderência à descrição textual. A principal premissa por trás deste sistema é a de que compreender bem a linguagem é mais importante para a geração de imagens de alta qualidade e para uma forte aderência ao *prompt* do que o tamanho do modelo de difusão (medido em termos do número de parâmetros treináveis da rede que estima o ruído). Assim, o sistema utiliza um codificador de texto T5-XLL com 11,3 bilhões de parâmetros para converter o texto do *prompt* em *embeddings*. Ele gera inicialmente imagens com 64×64 pixels, que são escaladas para 256×256 e posteriormente para 1024×1024 utilizando dois modelos de difusão para super-resolução.

Stable Diffusion [Rombach et al. 2022] realiza o processo de difusão em um espaço latente comprimido, reduzindo significativamente os custos computacional e de

memória. A técnica, conhecida como difusão latente (*latent diffusion*), consiste em comprimir as imagens utilizando um Autoencoder Variacional [Kingma and Welling 2014] e realizar o treinamento da rede neural utilizando esta representação comprimida. O treinamento é realizado no estilo DDPM. O processo de geração de imagens consiste então em partir de $z_T \sim \mathcal{N}(0, I)$ para obtenção de uma amostra $\hat{z}_0$, a qual é então mapeada para uma imagem utilizando o decodificador do Autoencoder Variacional. Ao utilizar uma representação comprimida no espaço latente, o modelo reduz tanto o tempo de treinamento quanto de amostragem.

DALL·E 3 [Betker et al. 2023], desenvolvido pela OpenAI, se caracteriza por uma ótima compreensão do *prompt* e uma forte aderência às instruções fornecidas pelo usuário. A premissa por trás do sistema DALL·E 3 é a de que modelos para geração de imagens a partir de texto podem ter seus desempenhos significativamente melhorados se treinados com imagens contendo descrições detalhadas. Assim, um sistema de descrição de imagens foi utilizado para produzir descrições detalhadas para o conjunto de treinamento. DALL·E 3 interpreta e reescreve os *prompts* dos usuários com maior detalhamento antes de gerar uma imagem. Esta funcionalidade é facilitada pela integração do sistema com os modelos de linguagem (LLMs) da OpenAI.

A Figura 2.7 mostra duas imagens geradas pelo sistema Imagen. A Figura 2.8 exibe imagens geradas com Stable Diffusion (esquerda) e DALL·E 3 (direita).

Figura 2.7. Imagens geradas com o sistema Imagen utilizando as descrições: (esquerda) "A forest with big trees and colorful leaves with a chicken next to it". (direita) "Três carros de corrida de fórmula 1".

2.6.4. O Impacto do Idioma Utilizado para Descrição da Imagem

A língua utilizada para a escrita do *prompt* pode ter um impacto significativo na imagem gerada. Assim, se um sistema que gera imagens a partir de comandos textuais foi treinado predominantemente utilizando pares de imagens e descrições em um dado idioma, ele tenderá a interpretar comandos neste idioma de forma mais precisa e a gerar imagens mais alinhadas com tais descrições do que utilizando *prompts* em outras línguas. Além disso, alguns conceitos, referências culturais e expressões idiomáticas podem ser melhor

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

Figura 2.8. Imagens geradas com Stable Diffusion (esquerda) e DALL·E 3 (direita) utilizando a descrição: "Fotografia de um homem em barco nas águas do rio Amazonas ao por do sol, cinemático.".

representadas em uma língua do que em outras. Para sistemas modernos como Imagen, DALL·E 3 e versões recentes de Stable Diffusion, descrições em português do Brasil tendem a produzir ótimos resultados, especialmente para conceitos comuns e descrições de cenas, como nos exemplos das Figuras 2.6 a 2.8. Em determinadas situações, a utilização de *prompts* em inglês pode resultar em maior aderência à descrição, visto que uma parcela considerável dos pares imagem-descrição disponíveis para treinamento de modelos de difusão encontram-se em inglês.

2.6.5. Condicionamento por Imagens e Condicionamentos Multimodais

Modelos de difusão podem utilizar várias modalidades de dados além de texto para guiar o processo de síntese de imagens. A princípio, qualquer modalidade que possa ser codificada na forma de *embeddings* que preservem informações semanticamente relevantes dos dados de entrada pode ser utilizada. Isso inclui, por exemplo, imagens, áudio e representações multimodais, em particular envolvendo texto e imagens.

Utilizando apenas imagem como *prompt*, ILVR (*Iterative Latent Variable Refinement*) [Choi et al. 2021] guia o processo de difusão em direção a uma imagem de referência injetando repetidamente informações de baixa frequência desta imagem durante o processo de difusão inversa. SDEdit [Meng et al. 2022] controla o processo de difusão em direção a uma imagem de referência x_0 acrescentando uma certa quantidade de ruído a esta com o intuito de eliminar os seus detalhes finos, mas preservando sua estrutura geral. A imagem resultante (e apenas parcialmente ruidosa) x_t é então usada em substituição a $x_T \sim \mathcal{N}(0, I)$ como ponto de partida para o processo de difusão reversa utilizado por DDPM/DDIM. ILVR e SDEdit assumem a existência de uma rede neural pre-treinada no estilo DDPM capaz de estimar a quantidade de ruído em uma imagem.

O condicionamento multimodal funciona de modo análogo ao descrito na Seção 2.6.1: as informações das diferentes modalidades são convertidas em *embeddings*, fundidas utilizando mecanismos de atenção cruzada, e reinjetadas nas camadas subsequentes

da rede neural ε_θ, influenciando os campos vetoriais da função escore condicionada. Isto faz com que as trajetórias definidas por estes campos vetoriais conduzam a amostras que tentam satisfazer as informações dos *prompts*. Neste caso, o modelo pode receber simultaneamente $c = (c_{\text{texto}}, c_{\text{imagem}}, c_{\text{áudio}}, \dots)$. Exemplos, incluem a geração de imagens a partir de texto + imagem, texto + áudio, geração de vídeo a partir de texto + imagem, etc. Exemplos recentes de modelos de difusão envolvendo texto + imagem utilizam filtragem para preservar a estrutura da imagem de entrada [Gu et al. 2024; Tamiosso et al. 2025].

2.7. Limitações e Questões Éticas

Modelos de difusão têm sido utilizados com sucesso em aplicações de diversas áreas, incluindo síntese de imagens e vídeos, criação e extração de modelos 3D a partir de imagens, aplicações médicas e design de novos produtos. Mas como todo modelo de aprendizagem de máquina, modelos de difusão apresentam vieses herdados de seus conjuntos de treinamento. Em geral, exigem grandes datasets e seus treinamentos costumam envolver alto custo computacional. Por se basearem em estatísticas de padrões visuais ao invés de em representações explícitas de objetos, modelos de difusão têm dificuldade com contagens e consistências estruturais (*e.g.*, frequentemente gerando uma quantidade de objetos diferente da solicitada, ou mãos com mais ou com menos de cinco dedos). Além dessas, outras limitações atuais são abordadas na Seção 2.8 que discute oportunidades de pesquisa. O impacto do idioma utilizado no treinamento do modelo foi tratado na Seção 2.6.4. Uma discussão de extrema relevância relacionada ao uso dessa tecnologia envolve **aspectos éticos** ligados ao seu elevado potencial para produção de *deep fakes*.

2.8. Oportunidades de Pesquisa

Apesar dos rápidos avanços observados em modelos de difusão, a área ainda apresenta inúmeros desafios, oferecendo diversas oportunidades de pesquisa. Alguns dos temas mais ativos no momento incluem: (i) A busca por **amostradores mais rápidos**, capazes de reduzir o tempo para geração de imagens sem degradar a qualidade dos resultados; (iii) **Condicionamento multimodal**: garantir condicionamento combinando diversas modalidades como texto, imagens, áudio, vídeos, modelos 3D, mapas de profundidade, etc.; (iii) **Geração de vídeos** garantindo consistência temporal, movimentação natural de câmera, produção de vídeos longos, e a redução do custo computacional para sua geração; (iv) **Aspectos físicos do mundo real**: atualmente os modelos aprendem relações entre elementos das cenas com base na regularidade estatística das imagens no conjunto de treinamento. Por exemplo, o gato da Figura 2.6 está em contato com a cadeira não devido ao conhecimento sobre a lei da gravidade, mas porque essa relação aparece sistematicamente no conjunto de treinamento; e (v) **Geração de cenas 3D**: ao invés de apenas imagens 2D.

Apesar do enorme sucesso alcançado por modelos de difusão, várias questões teóricas permanecem em aberto. Por exemplo, quais são os limites fundamentais de processos generativos baseados na função escore?

2.9. Conclusão

Este tutorial apresentou uma introdução a modelos de difusão para geração de imagens. Os princípios matemáticos e estatísticos que garantem a correção desta classe de técnicas

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

e a obtenção de imagens de alta qualidade foram discutidos de forma intuitiva, porém tecnicamente rigorosa. Modelos de difusão utilizam a função escore para definir trajetórias que levam amostras consistindo de puro ruído para as regiões mais prováveis em uma distribuição empírica de probabilidades definida pelo conjunto de imagens de treinamento. A função escore, por sua vez, pode ser aproximada por uma rede neural ε_θ que estima o ruído acrescentado a amostras do conjunto de treinamento. O condicionamento de um modelo de difusão por meio de *prompts*, especialmente quando combinado com *classifier-free guidance* permite redefinir os campos vetoriais da função escore de modo a direcionar amostras para regiões do espaço de imagens mais alinhadas com as descrições dos *prompts*. Isto permite aos modelos de difusão gerar imagens de alta qualidade e alinhadas com as instruções dadas pelos usuários. Apesar do rápido avanço experimentado, modelos de difusão apresentam inúmeros desafios e oportunidades de pesquisa.

Créditos das imagens

As imagens nas Figuras 2.2 a 2.5 foram utilizadas de acordo com licença CC. A foto na Figura 2.2 (esquerda) é de mosher13; As fotos na Figura 2.3 são de - flores: mosher13, iGlacier e MadalenaPestana; gatos: nist6ss, Jon_a_ross e Mike_Knell; cães: basykes, cfiesler e twodolla. As fotos na Figura 2.5 são de - flor: FotoGuy$_4$9057; $gato : El\_C$; $cão : eXploration\_Etoile$.

Referências

[Abramson et al. (2024)] Josh Abramson, Jonas Adler, et al. Accurate structure prediction of biomolecular interactions with alphafold 3. *Nature*, 630(8016):493–500, 2024.

[Alibaba (2026)] Alibaba. Wan video. AI video generator, 2026. URL `https://wan.video/`. Last Access, May 2026.

[Betker et al. (2023)] James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn.openai.com/papers/dall-e-3.pdf*, 2(3):8, 2023.

[Black Forest (2026)] Labs Black Forest. Flux. AI image generator, 2026. URL `https://www.fluxpro.ai`. Last Access, May 2026.

[Choi et al. (2021)] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. In *Proceedings of ICCV*, pages 14367–14376, 2021.

[Deevid AI (2026)] Deevid AI. Deevid. AI video generator, 2026. URL `https://deevid.ai/`. Last Access, May 2026.

[Google DeepMind (2025)] Google DeepMind. Introducing nano banana pro. `https://blog.google/innovation-and-ai/products/nano-banana-pro/`, November 2025. Google Blog, November 20, 2025. Last Access, May 2026.

[Gu et al. (2024)] Zeqi Gu, Ethan Yang, and Abe Davis. Filter-guided diffusion for controllable image generation. In *ACM SIGGRAPH 2024 conference papers*, pages 1–10, 2024.

[He et al. (2016)] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of CVPR*, pages 770–778, 2016.

[Heusel et al. (2017)] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 30, 2017.

[Ho et al. (2020)] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *NeurIPS*, 33:6840–6851, 2020.

[Kingma and Welling (2014)] Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd Intern. Conf. on Learning Representations, ICLR 2014*, 2014.

[Kuaishou (2026)] Technology Kuaishou. Klingai. AI video generator, 2026. URL `https://kling.ai/`. Last Access, May 2026.

[Luma Labs (2026)] Luma Labs. Luma dream machine. AI video generator, 2026. URL `https://lumalabs.ai/`. Last Access, May 2026.

[Luo and Hu (2021)] Shitong Luo and Wei Hu. Diffusion probabilistic models for 3d point cloud generation. In *Proceedings of CVPR*, pages 2837–2845, 2021.

[Meng et al. (2022)] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL `https://openreview.net/forum?id=aBsCjcPu_tE`.

[Midjourney, Inc. (2026)] Midjourney, Inc. Midjourney. AI image generator, 2026. URL `https://www.midjourney.com`. Last Access, May 2026.

[OpenAI (2026)] OpenAI. Sora 2. AI video generator, 2026. URL `https://openai.com/index/sora-2/`. Last Access, May 2026.

[Peebles and Xie (2023)] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of ICCV*, pages 4195–4205, 2023.

[Rombach et al. (2022)] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of CVPR*, pages 10684–10695, 2022.

[Ronneberger et al. (2015)] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Intern. Conf. on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[Runway AI (2026)] Inc Runway AI. Runway gen 4.5. AI video generator, 2026. URL `https://runwayml.com/`. Last Access, May 2026.

[Saharia et al. (2022)] Chitwan Saharia, William Chan, et al. Photorealistic text-to-image diffusion models with deep language understanding. *NeurIPS*, 35:36479–36494, 2022.

[Sohl-Dickstein et al. (2015)] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pages 2256–2265. pmlr, 2015.

[Song et al. (2021)] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021. URL `https://openreview.net/forum?id=St1giarCHLP`.

[Song and Ermon (2019)] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *NeurIPS*, 32, 2019.

[Song et al. (2020)] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[Tamiosso et al. (2025)] Gustavo Lopes Tamiosso, Caetano Müller, Lucas Spagnolo Bombana, and Manuel M Oliveira. Memory-efficient filter-guided diffusion with domain transform filtering. *Computers & Graphics*, page 104389, 2025.

[Vaswani et al. (2017)] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is All You Need. *NeurIPS*, 30, 2017.

[Wang et al. (2025)] Chen Wang, Hao-Yang Peng, Ying-Tian Liu, Jiatao Gu, and Shi-Min Hu. Diffusion models for 3d generation: A survey. *Computational Visual Media*, 11 (1):1–28, 2025.

Capítulo

3

Desenvolvimento de Software Centrado No Usuário: Construindo produtos com b_thinking, UX e Agile

Raul Sidnei Wazlawick (UFSC), Ranieri Alves dos Santos (UFSC e UNISUL), Jades Fernando Hammes (UFSC) e Célio Luiz Cunha (UFSC)

Abstract

This chapter presents the trajectory and methodological maturity of the Bridge Laboratory (UFSC) in the creation of huge digital products for public management. The work is based on the philosophy of Chadō (Way of Tea), which preaches the mastery of serving with empathy and mindfulness, transposing this lesson to the development of systems that solve people's real pains. The core of the chapter is the b_thinking model, a framework that integrates UX (User Experience), Design Thinking, Lean UX, and agile approaches. This model is supported by principles such as the formation of small cross-functional teams, the focus on results without waste, shared knowledge, and the strategy of testing hypotheses before scaling production.

Resumo

Este capítulo apresenta a trajetória e a maturidade metodológica do Laboratório Bridge (UFSC) na criação de grandes produtos digitais para a gestão pública. A obra é fundamentada na filosofia do Chadō (Caminho do Chá), que prega a maestria em servir com empatia e atenção plena, transpondo essa lição para o desenvolvimento de sistemas que resolvam as dores reais das pessoas. O cerne do capítulo é o modelo b_thinking, uma estrutura que integra as abordagens de UX (User Experience), Design Thinking, Lean UX e metodologia ágil. Esse modelo é sustentado por princípios como a formação de equipes multifuncionais pequenas, o foco no resultado sem desperdícios, o conhecimento compartilhado e a estratégia de testar hipóteses antes de escalar a produção.

Este capítulo foi realizado com recursos do CNPq Processo: 303.250/2023-2 e Ministério da Saúde TED 92/2021. A ferramenta de inteligência artificial NotebookLM,foi empregada na revisão textual deste trabalho. O capítulo consiste em um resumo estendido do livro de Wazlawick, *et al*. [2025].

3.1. Introdução

Na cerimônia do chá, a maestria do anfitrião não reside apenas no domínio técnico dos utensílios ou das infusões, mas no ato de servir com atenção plena e empatia, tratando a interação como um momento único de alívio aos anseios do convidado.

Para o desenvolvimento de software, a lição é direta: a tecnologia de ponta e os frameworks sofisticados perdem sua serventia se não houver compreensão real das necessidades do usuário. O desenvolvimento centrado no usuário exige buscar essa "atenção plena", transformando códigos e interfaces em produtos que resolvam problemas humanos de forma empática.

3.1.1 A Trajetória de Excelência do Laboratório Bridge

O Laboratório Bridge, sediado na Universidade Federal de Santa Catarina (UFSC), consolidou-se como um centro de referência em tecnologia para a gestão pública. Sua história está intrinsecamente ligada à informatização da saúde pública no Brasil, iniciada em 2011 com o planejamento da Estratégia e-SUS AB junto ao Ministério da Saúde.

Em 2013, o laboratório assumiu oficialmente o desenvolvimento de sistemas críticos, iniciando com uma equipe de 17 especialistas (professores, contratados CLT e bolsistas). Desde então, o Bridge entregou mais de 55 produtos, gerenciando dados clínicos de milhões de brasileiros e expandindo sua atuação para áreas como educação (MEC e FNDE) e vigilância sanitária (ANVISA). Em 2026, a organização conta com mais de 230 colaboradores, mantendo seu DNA universitário aliado a uma cultura de alta performance.

3.1.2 Valores Culturais: O Jeito *Bridger* de Ser

A cultura do laboratório é sintetizada em cinco pilares fundamentais que guiam o comportamento das equipes:

- *Ser a diferença*: Orgulho em usar a tecnologia para melhorar vidas.
- *Fazer bem-feito, inclusive o café*: Busca pela agilidade sem renunciar à excelência.
- *Explorar novos mundos*: Estímulo à proatividade e inovação constante.
- *Conectar pessoas a momentos*: Valorização das conexões humanas dentro dos times.
- *Compartilhar dá mais XP*: A crença de que a troca de conhecimento potencializa os resultados coletivos.

3.1.3 Desmistificando a UX (*User Experience*)

Uma das missões deste capítulo é esclarecer que a Experiência do Usuário (UX) não é uma disciplina restrita a designers gráficos ou *enfeitadores* de telas. Ela é compreendida como a área do conhecimento baseada na ergonomia, arquitetura de informação e design para projetar produtos voltados ao pleno uso de seres humanos.

A UX é o elo de satisfação entre o produto, quem o projeta e o usuário, indo muito além dos aspectos visuais para garantir que o sucesso do usuário seja atingido.

3.1.4 Termos-Chave e Distinções Importantes

Para garantir uma linguagem comum ao longo do texto, definem-se quatro distinções cruciais:

- *Design*: Não deve ser lido apenas como estética, mas como o processo e a sequência de eventos necessários para a criação de um objeto ou solução. O designer é quem projeta o produto digital.
- *Usuário vs. Cliente*: O usuário é o indivíduo que interage diretamente com o sistema no cotidiano para realizar tarefas. O cliente é aquele que representa o negócio, toma decisões de compra e define benefícios e custos, embora nem sempre opere o software.
- *Produto vs. Sistema*: Embora no meio acadêmico existam distinções técnicas, o capítulo utiliza esses termos de forma intercambiável para representar o resultado dos esforços de desenvolvimento, seja ele um aplicativo, portal, jogo ou serviço.

O foco do capítulo é compartilhar a maturidade adquirida pelo Laboratório Bridge em mais de uma década, oferecendo um modelo prático (*b_thinking*) para que outros times possam criar produtos que sejam não apenas funcionais, mas amados por quem realmente irá utilizá-los.

3.2 Desenvolvendo Software com Foco no Usuário

Embora a tecnologia esteja onipresente no cotidiano profissional e pessoal, a existência de software que não atinge seus objetivos ainda é um problema comum. Muitos produtos falham não por deficiências técnicas de codificação, mas por falta de comunicação, atrasos crônicos e, principalmente, baixa satisfação do usuário. Quando um sistema não atende plenamente às necessidades de quem o opera, ele tende a ser rejeitado ou utilizado de forma comprometida, o que prejudica o valor percebido pelo cliente.

Software difícil de usar gera um *efeito bola de neve* negativo para a organização: aumenta a demanda por suporte técnico, exige treinamentos mais longos e custosos e pode levar ao abandono total da solução. Nesse contexto, a grande métrica de sucesso de um produto digital deixa de ser apenas a sua operação técnica correta e passa a ser a satisfação plena do usuário final.

3.2.1 Engenharia de Software Centrada no Usuário

A Engenharia de Software evoluiu de um foco estrito na máquina para uma abordagem orientada ao ser humano. Essa trajetória é dividida em marcos fundamentais:

- 1940-1950: O foco era quase exclusivamente em conexões físicas e lógica de máquina, sem a existência de *software* como o compreendemos hoje.
- 1960-1970: Surgem as linguagens de alto nível (como COBOL e Fortran) e os primeiros modelos de processo para enfrentar a chamada "Crise do Software", sendo o Modelo Cascata a principal referência da época.
- 1980-1990: A indústria buscou a "bala de prata" através de ferramentas CASE e da consolidação da Orientação a Objetos, focando na modularização e reutilização de código.
- *Anos* 2000: A explosão da Metodologia Ágil trouxe um novo paradigma, priorizando indivíduos e interações sobre processos e ferramentas rígidas.

A partir de 2010, consolidou-se a visão de que não basta entregar um software funcional e ágil; ele deve ser centrado no usuário (UX). A UX tornou-se o diferencial essencial para a fidelização de clientes, unindo usabilidade e prazer no uso.

3.2.2 O Processo de UX no Laboratório Bridge

O Laboratório Bridge percebeu, ao longo de sua atuação em projetos de grande escala nacional, que a diversidade de contextos — desde grandes centros urbanos até aldeias indígenas isoladas — exigia uma abordagem de design robusta e adaptável. Inicialmente, as práticas de UX (entrevistas, testes e métricas) eram aplicadas de forma heterogênea por cada equipe, o que gerava ganhos isolados, mas dificultava a padronização da qualidade.

Diante da necessidade de garantir transparência e eficiência em sistemas complexos de gestão pública, o laboratório empenhou-se em padronizar o conhecimento de UX. Esse esforço resultou na criação do *b_thinking*, um modelo que não é apenas um guia de métodos, mas uma estrutura cultural para construir o *caminho do chá* do desenvolvimento de software.

3.2.3 O Nascimento do *b_thinking*

O modelo *b_thinking* foi desenvolvido após visitas de *benchmarking* a grandes empresas de tecnologia e um mergulho nas práticas de Lean UX, Design Thinking e metodologia ágil. Ele foi projetado para ser:

- *Colaborativo*: Focando no compartilhamento de conhecimento entre designers, desenvolvedores e PO's.
- *Iterativo*: Permitindo que o produto seja lapidado a cada ciclo de descoberta e entrega.
- *Flexível*: Capaz de se adaptar às particularidades de cada projeto, mantendo sempre a experiência do usuário como norte.

O *b_thinking* é o cerne da maturidade metodológica do Laboratório Bridge, permitindo que as mais de quinze equipes multifuncionais trabalhem em harmonia para transformar problemas complexos em soluções digitais que os usuários realmente desejem utilizar. O sucesso de um software, portanto, depende de processos que valorizem o aprendizado contínuo e a empatia profunda com as dores de quem será servido pela tecnologia.

3.3 Modelo de Processo Centrado no Usuário

O modelo *b_thinking* é fruto de mais de dez anos de experiência do Laboratório Bridge no desenvolvimento de produtos para o governo eletrônico brasileiro.

Ele não é um guia rígido, mas uma estrutura adaptável que combina abordagens de *User Experience* (UX), a sensibilidade do *Design Thinking*, a agilidade do *Lean UX* e a execução de métodos ágeis, como Scrum e Kanban. O objetivo central é oferecer uma abordagem descritiva para solucionar problemas complexos, garantindo que o produto atenda às necessidades reais dos usuários.

3.3.1 Os Quatro Princípios Norteadores

O sucesso do modelo é sustentado por quatro pilares fundamentais que guiam as equipes no dia a dia:

- *Equipes multifuncionais, mas pequenas*: O modelo defende times reduzidos para diminuir a complexidade da comunicação e evitar gargalos. Essas equipes mesclam diferentes habilidades (negócios, design e desenvolvimento), permitindo autonomia e respostas rápidas a mudanças.
- *Sem desperdícios, foco no resultado*: Inspirado no Lean UX, este princípio foca na construção enxuta, removendo atividades que não agregam valor ao usuário. O cerne aqui é a entrega constante de MVPs (Produtos Mínimos Viáveis) baseados em necessidades reais.
- *Conhecimento compartilhado*: Visa alinhar todos os membros do time em torno de um objetivo comum, evitando *ilhas de conhecimento*. A colaboração contínua garante que desenvolvedores e designers compreendam as dores do usuário de forma sistêmica.
- *Testar antes, escalar depois*: Antes de investir recursos massivos em desenvolvimento, o modelo prioriza a validação de hipóteses em pequena escala através de protótipos testáveis. Isso permite *errar barato* e ajustar o produto antes da produção definitiva.

3.3.2 A Estrutura Cíclica (Discovery e Delivery)

O processo *b_thinking* é dividido em duas macroetapas integradas que formam um ciclo contínuo (Figura 3.1): Descoberta (*Discovery*) e Entrega (*Delivery*). A transição entre elas é fluida, permitindo que a equipe retorne a etapas anteriores sempre que novos aprendizados surgirem durante a validação ou construção.

3.3.2.1 Macro etapa de Descoberta (Discovery)

Focada em "fazer a coisa certa", esta fase busca compreender profundamente o problema e identificar as necessidades dos usuários por meio de UX Research. Suas etapas incluem:

- *Briefing*: Alinhamento inicial de escopo, prazos, stakeholders e público-alvo.
- *Imersão*: Pesquisa de campo e documental. Utiliza a Matriz CSD (Certezas, Suposições e Dúvidas) para organizar o conhecimento e guiar a investigação.
- *Definição*: Síntese das descobertas em artefatos como Personas, Mapas de Empatia e Canvas de Proposta de Valor para guiar o design da solução.
- *Ideação*: Geração de soluções criativas através de dinâmicas como Brainstorming, *Crazy* 8s e Seis Chapéus do Pensamento.
- *Prototipação de Ideias*: Tangibilização das soluções em modelos de baixa ou média fidelidade (*rabiscoframes* e *wireframes*) para validar a lógica sem compromisso com a estética final.

3.3.2.2 Macro etapa de Entrega (Delivery)

Focada em "fazer certo a coisa", transforma as ideias validadas em software funcional. Suas etapas são:

- *Validação de Ideias*: Testes de usabilidade com usuários reais para confirmar se a proposta resolve o problema.
- *Delimitação*: Definição do escopo do MVP e priorização das funcionalidades através de técnicas como MoSCoW ou Matriz de Impacto *vs*. Esforço (Figura 3.2).

- *Refinamento*: Detalhamento técnico de regras de negócio, protótipos de alta fidelidade e critérios de aceitação para preparar a entrada nas Sprints.
- *Sprint*: Ciclos ágeis de codificação e testes, onde o software é construído incrementalmente.
- *Avaliação da Release*: Coleta de feedbacks e métricas de satisfação (como SUS e NPS) após o lançamento para medir o sucesso da entrega.
- *Melhoria Contínua*: O ciclo nunca termina; os aprendizados do uso real alimentam novas hipóteses e evoluções do produto.

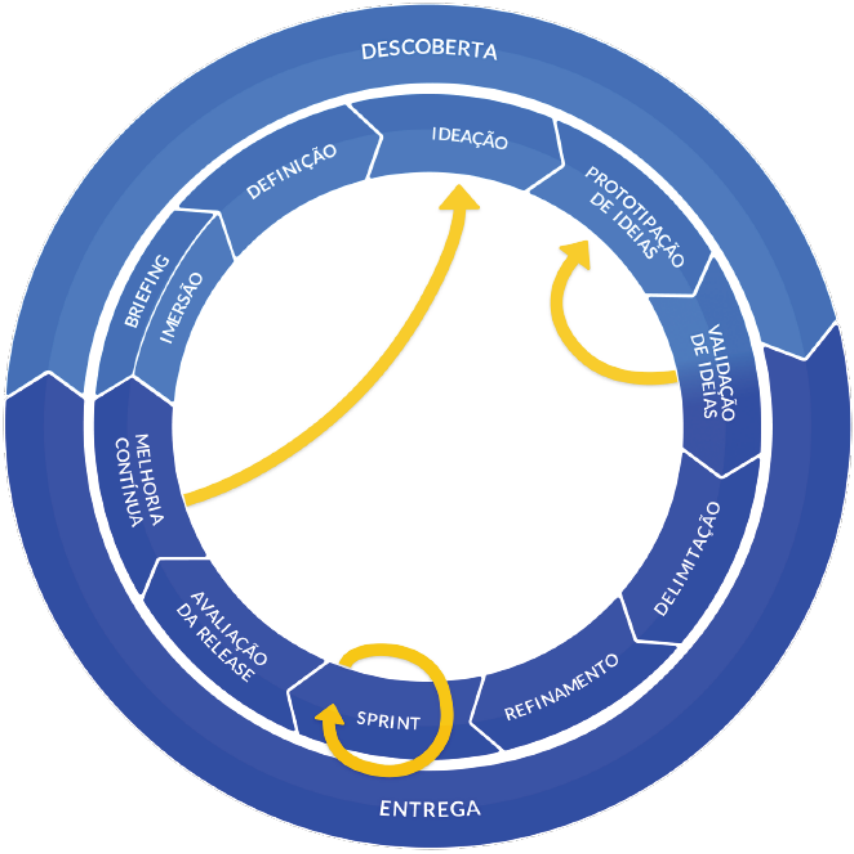

Figura 3.1. Ciclo de Etapas do b_thinking

3.3.3 Integração com *Privacy by Design*

O *b_thinking* incorpora nativamente os sete princípios do *Privacy by Design*, garantindo que a proteção de dados seja considerada desde a concepção do projeto. O modelo foca em ser proativo, ter a privacidade como padrão (*default*) e garantir a segurança de ponta a ponta, alinhando o desenvolvimento às exigências da LGPD BRASIL [2018].

Essa sequência de etapas é uma sugestão baseada na maturidade do Laboratório Bridge, mas cada equipe deve adaptá-la conforme a complexidade e necessidade do projeto, mantendo sempre o usuário no centro das decisões.

3.4 Construção de um Produto

Esta seção apresenta resumidamente um estudo de caso real do desenvolvimento de um módulo de monitoramento de pacientes com tuberculose utilizando o modelo b_thinking.

Figura 3.2: Matriz de Impacto *vs.* Esforço

O projeto nasceu da necessidade de informatizar a vigilância epidemiológica em um município de médio porte, onde o acompanhamento de pacientes com tuberculose era feito de forma fragmentada através de fichas de papel e planilhas eletrônicas. O objetivo era centralizar o cuidado no prontuário eletrônico para evitar retrabalho e perdas de informações críticas para o tratamento, que dura no mínimo seis meses. A equipe de Discovery foi composta por um Product Owner (Rafael), uma Designer (Juliana) e um Analista (Jairo). O processo iniciou com o Briefing, alinhando com a gerência prazos (34 semanas) e objetivos, como a geração de alertas automáticos e a integração com sistemas do Ministério da Saúde.

3.4.1 Imersão e Pesquisa (UX Research)

Para compreender as dores reais, a equipe utilizou diversas ferramentas:

- *Matriz CSD*: Organizou certezas, suposições e dúvidas iniciais, sendo atualizada ao longo do projeto.

- *Observação de Campo*: Realizada em uma Unidade Básica de Saúde, permitiu identificar que o excesso de burocracia manual deixava os profissionais sentindo-se desorganizados.
- *Entrevistas e Benchmarking*: Conversas com médicos e a análise de quatro sistemas similares no mercado ajudaram a mapear funcionalidades essenciais e falhas de usabilidade em soluções existentes.

3.4.2 Definição e Ideação

Os dados coletados foram sintetizados na Persona Marina Velasco (Figura 3.3), uma enfermeira que representa o perfil majoritário dos usuários. Através do Mapa de Empatia e do Canvas de Proposta de Valor (Figura 3.4), a equipe focou em soluções que reduzissem a carga cognitiva do profissional.

A técnica MoSCoW priorizou como "Must have" a visualização clara do paciente e o registro de exames. Na ideação, dinâmicas como o Brainstorming e os Seis Chapéus do Pensamento exploraram ideias como a teleconsulta, que foi avaliada criticamente sob a ótica de riscos e benefícios.

3.4.3 Prototipação e Validação Inicial

A designer criou *Rabiscoframes* (baixa fidelidade - Figura 3.5) para validar a arquitetura da informação rapidamente. Após ajustes, desenvolveu o protótipo de alta fidelidade no Figma utilizando o Bold Design System (bold.bridge.ufsc.tech/pt/). Foram realizados Testes de Usabilidade com cinco usuários, resultando em uma pontuação SUS de 74,2 (considerada boa) e a identificação de melhorias necessárias na tela de resumo, validadas via mapas de calor.

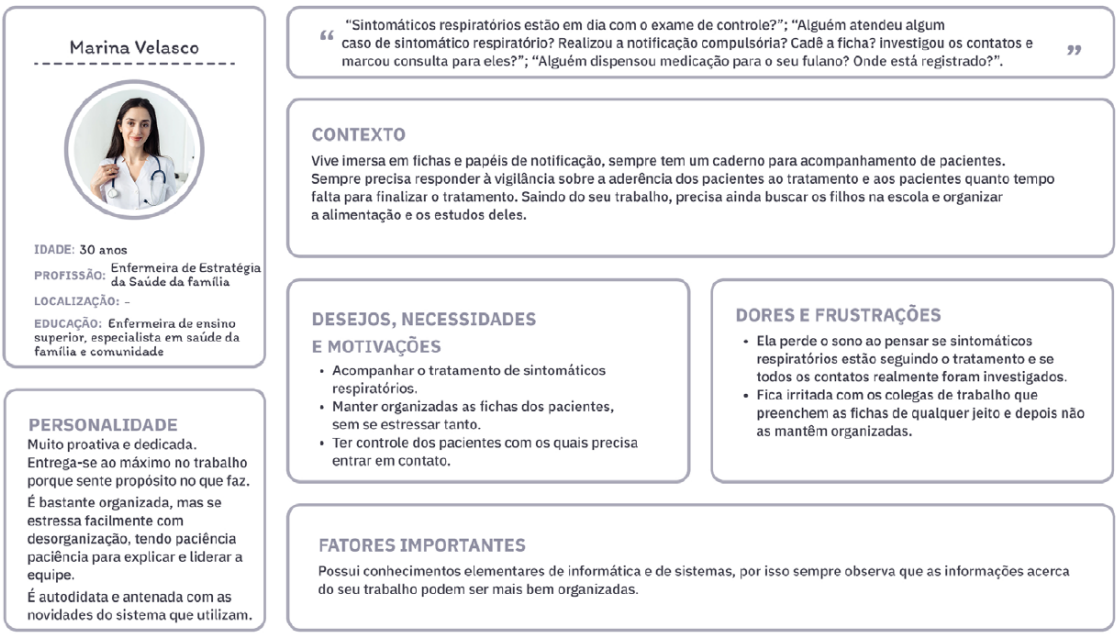

Figura 3.3: Persona: Marina Velasco

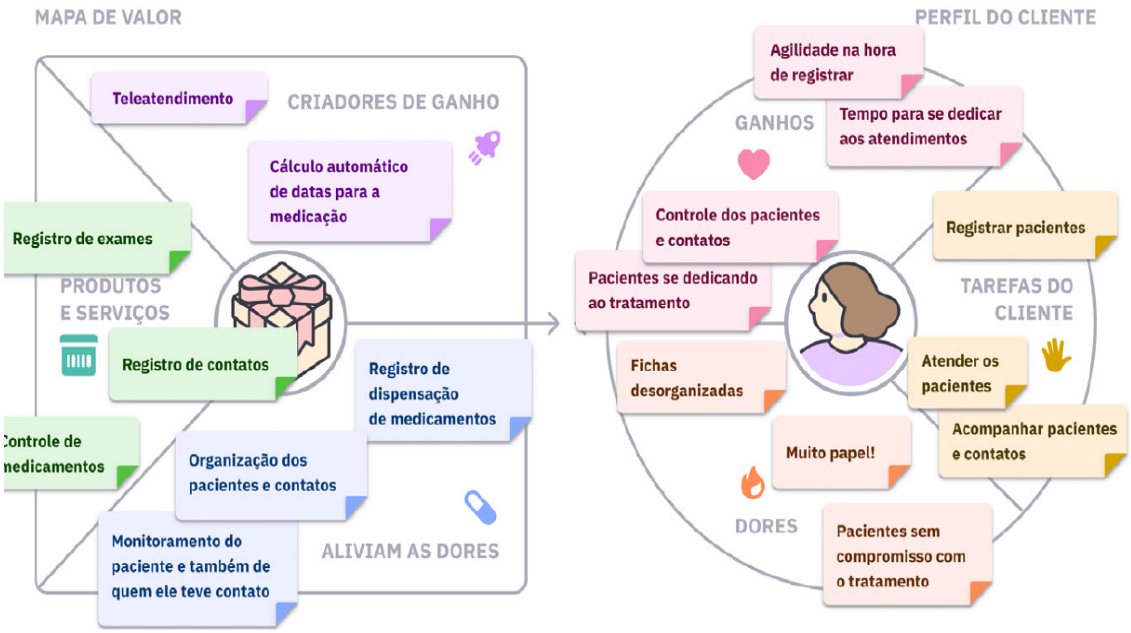

Figura 3.4: Canvas de Proposta de Valor

3.4.5 Delimitação do MVP e Planejamento

A etapa de Delimitação definiu o escopo do MVP (Produto Mínimo Viável) para ser construído em dezoito semanas, divididas em nove sprints de duas semanas cada. Foram aplicadas as seguintes técnicas de priorização técnica e de negócio:

- *Planning Poker* (*T-shirt Sizing*): O time estimou o esforço de cada funcionalidade em tamanhos (P, M, G e GG), identificando que a estrutura do módulo no prontuário era a tarefa mais complexa.
- *Buy-a-feature*: Uma dinâmica onde stakeholders "compravam" funcionalidades com um orçamento fictício, o que confirmou o alto valor percebido no registro de exames e acompanhamento de contatos.

3.4.6 Refinamento e Execução (Sprints)

No Refinamento, o PO Rafael detalhou as Histórias de Usuário com critérios de aceitação rigorosos para guiar os desenvolvedores. Durante as Sprints, a equipe utilizou o Quadro Kanban para gerenciar visualmente o fluxo de trabalho (*To Do*, *Doing*, *Testing* e *Done*) e realizou *Dailies* para alinhar o progresso diário. As retrospectivas, como a dinâmica do "Balão de Ar", ajudaram a identificar gargalos no processo e promover a melhoria contínua do time.

3.4.7 Avaliação da Release e Lançamento

Após a conclusão dos ciclos de desenvolvimento e testes de QA, o sistema foi implantado em uma unidade piloto. A Avaliação da Release envolveu a coleta de feedbacks reais de uso:

- *Métricas de Sucesso*: O produto atingiu um NPS (Net Promoter Score) de 91, situando-se na Zona de Excelência de satisfação do usuário.

- *Feedback Qualitativo*: Profissionais relataram que o sistema trouxe maior agilidade e organização ao tratamento, eliminando a dependência de papéis.

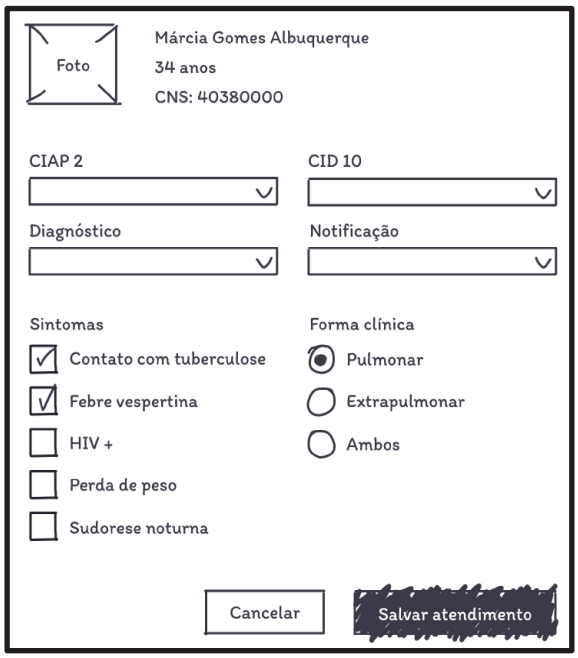

Figura 3.5: Protótipo de baixa fidelidade

3.4.8 Conclusão e Melhoria Contínua

O processo de software centrado no usuário é cíclico. Mesmo após o sucesso do lançamento, a equipe mantém o foco na Melhoria Contínua, monitorando o uso real para propor novas evoluções e ajustes baseados em novas hipóteses e dores que surjam com a maturidade do produto no mercado.

3.5 Métodos e Ferramentas de Briefing e Imersão

A partir deste capítulo, vamos detalhar os métodos e ferramentas do b_thinking para auxiliar no processo de construção de produtos com foco no usuário.

3.5.1 Alinhamento com o Gerente

O rumo de um projeto de UX é definido logo em seus momentos iniciais. O objetivo do alinhamento com supervisores ou gerentes é identificar informações cruciais como o escopo, objetivos gerais, restrições de projeto e os prazos de entrega. Um alinhamento bem executado reduz drasticamente as chances de retrabalho futuro, pois garante clareza no repasse das tomadas de decisão e nas expectativas de resultados.

3.5.2 Análise de Impacto

A análise de impacto é apresentada através da metáfora de um quebra-cabeças, onde cada peça de software possui dependência em relação a outras. Esta ferramenta serve para identificar e documentar as consequências de alterar ou incluir novas funcionalidades. A análise foca em três pilares:

- *Dependência*: Identifica quais partes devem ser retrabalhadas ao implementar alterações.
- *Consequências potenciais*: Avalia os riscos associados ao desenvolvimento e recursos extras.
- *Rastreabilidade*: Analisa o que precisa ser alterado em diferentes níveis de documentação e no próprio código.

3.5.3 Benchmarking

O objetivo desta técnica é avaliar produtos e serviços de concorrentes ou similares para aumentar a competitividade e melhorar os resultados do próprio produto. O texto resgata o marco histórico do uso de testes militares no século XIX para analisar armas de fogo (*bench*). O processo de benchmarking no b_thinking envolve a definição de itens a serem comparados, a escolha de similares e o estabelecimento de indicadores para comparação em uma matriz ou tabela, facilitando a visualização de pontos fortes e fracos.

3.5.4 Briefing com o Cliente

Diferente do alinhamento gerencial, o briefing com o cliente busca alinhar o problema inicial sob a ótica de quem demanda a solução. Ele auxilia na aquisição de informações sobre o público-alvo, funcionalidades esperadas, cenário de uso, prioridades e fontes de informação como manuais ou portarias. O artefato resultante serve como um guia para as etapas subsequentes de descoberta.

3.5.5 Desk Research

Também chamada de pesquisa secundária ou documental, a Desk Research fornece uma visão ampla da área estudada sem a necessidade inicial de trabalho de campo. Ela otimiza recursos ao utilizar dados já disponíveis na organização ou em fontes externas, como órgãos do governo (IBGE, portarias) e Internet. Uma técnica recomendada para analisar esses dados é o "What's on your radar?", que organiza os itens coletados por proximidade e relevância ao objetivo do projeto.

3.5.6 Entrevista

A entrevista é um instrumento de coleta de dados qualitativos baseado em conversas que buscam identificar comportamentos e dores dos usuários. Sugere-se roteiros semiestruturados e atenção de dicas valiosas para evitar vieses, como: fugir de perguntas que induzam respostas "sim" ou "não", incentivar o relato de experiências e não mostrar protótipos cedo demais para não influenciar a opinião do entrevistado.

3.5.7 Matriz CSD (Certezas, Suposições e Dúvidas)

Esta é uma das ferramentas mais icônicas do Discovery, utilizada para organizar o conhecimento do time. Ela é dividida em três colunas:

- *Certezas*: Fatos e dados consensuais.
- *Suposições*: Hipóteses levantadas que precisam de validação.
- *Dúvidas*: Interrogações sobre o problema que o time ainda desconhece.

A Matriz CSD (Figura 3.6) é considerada uma ferramenta "viva", sendo atualizada à medida que as dúvidas se transformam em certezas durante a pesquisa.

3.5.8 Observação

Por fim, o capítulo aborda as técnicas de observação para entender como os usuários se comportam em seu contexto natural. O laboratório utiliza variações como:

- *Shadowing* (sombra): Acompanhamento atento dos processos do usuário.
- *Fly on the wall*: Observação passiva, sem intervenção do pesquisador.
- *Cliente Oculto*: O pesquisador se passa por um agente para interagir no processo.
- *Etnografia*: Imersão cultural profunda para entender valores e detalhes da comunidade.

Um caso de observação em uma Unidade Básica de Saúde Indígena, onde a equipe identificou problemas de fragmentação e burocracia manual mostrou que estes não seriam detectados apenas em entrevistas de escritório.

3.6 Métodos e Ferramentas de Ideação

Esta seção detalha como o Laboratório Bridge utiliza a criatividade estruturada para transformar dados de pesquisa em soluções inovadoras e centradas no ser humano.

3.6.1 Introdução à Ideação

A ideação é a fase da jornada de UX onde a equipe explora sua capacidade criativa para gerar soluções que atendam aos desafios identificados nos momentos de briefing e imersão. O objetivo principal é garantir que o trabalho seja assertivo, diminuindo incertezas por meio da geração de hipóteses de solução.

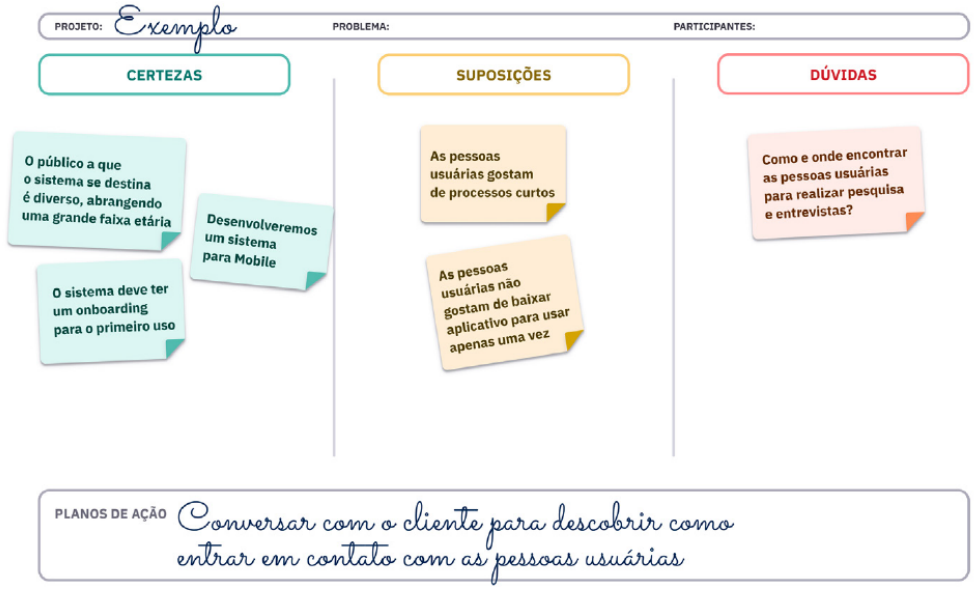

Figura 3.6: Matriz CSD

3.6.2 Brainstorming e suas Variações

O Brainstorming (tempestade cerebral) é uma técnica clássica para explorar a criatividade individual ou em grupo. Introduzida por Osborn [1970], ela é regida por quatro regras

básicas: não criticar ideias alheias, privilegiar a quantidade, incentivar ideias "fora da caixa" e permitir a combinação de propostas para aprimorá-las:

- *Brainwriting*: Similar ao brainstorming, mas foca na escrita. Os participantes anotam ideias e as passam para o colega ao lado, que deve complementá-las, garantindo que todos contribuam e evitando a inibição de membros mais tímidos.
- *Cinco Porquês*: Técnica voltada para encontrar a causa raiz de um problema através do questionamento repetitivo.
- *Mapas Mentais*: Diagramas que auxiliam no mapeamento de fluxos de informação e pensamentos interconectados, facilitando a compreensão sistêmica.

3.6.3 *Crazy* 8

O *Crazy* 8 é uma dinâmica de ideação rápida que desafia cada participante a esboçar oito ideias diferentes em oito minutos. Seu objetivo é superar a primeira ideia óbvia e estimular soluções inovadoras. O processo ocorre em quatro etapas: apresentação do problema, explicação técnica, execução física (preferencialmente em papel para evitar bloqueios digitais) e votação das melhores propostas pelo grupo.

3.6.4 Personas: Humanizando o design

As Personas são representações fictícias de grupos de usuários baseadas em dados reais coletados na imersão. Elas sintetizam comportamentos, necessidades e frustrações, servindo como guia para que o time projete funcionalidades que realmente façam sentido para o ser humano que utilizará o produto. No Laboratório Bridge, a construção de personas auxilia a equipe a sentir empatia e prever tomadas de decisão sob a perspectiva do usuário.

3.6.5 Geração de Hipóteses: Double diamond

O modelo *Double Diamond* (Diamante Duplo) organiza o processo de ideação em ciclos de divergência e convergência. Ele é dividido em quatro fases: Descobrir (pesquisa), Definir (foco no desafio), Desenvolver (ideação de soluções) e Entregar (testes e validação). Outra ferramenta é a *Opportunity Solution Tree*, que mapeia soluções conectando-as aos objetivos desejados.

3.6.6 Tarot da Tecnologia: Projeções éticas

Diferente das ferramentas de design convencionais, o Tarot da Tecnologia busca instigar o time a pensar nas consequências éticas e sociais do produto a longo prazo. O baralho é composto por doze cartas com perguntas provocativas sobre inclusão, exclusão e impactos não intencionais da tecnologia na sociedade.

3.6.7 Exercícios Criativos: Teste dos trinta círculos

Utilizado como aquecimento, o Teste dos Trinta Círculos desafia os participantes a transformarem círculos em branco em objetos reconhecíveis em apenas três minutos. É uma ferramenta de "ginástica mental" que exercita a velocidade de raciocínio e a quebra de padrões antes de sessões complexas de ideação.

3.6.8 Seis Chapéus do Pensamento

Esta técnica é baseada no método de Bono [1999]:

- *Azul*: Gestão e controle do processo.

- *Branco*: Fatos, dados e neutralidade.
- *Amarelo*: Otimismo e busca por benefícios.
- *Preto*: Julgamento crítico, riscos e cautela.
- *Vermelho*: Intuição, emoções e sentimentos.
- *Verde*: Criatividade e geração de alternativas inovadoras.

Assim, cada participante da reunião literalmente veste um chapéu e dialoga com os demais sempre de acordo com o viés de pensamento proposto pelas cores.

3.6.9 Análise de Tarefas (HTA)

A *Hierarchical Task Analysis* (HTA) é utilizada para compreender a sequência de ações que o usuário executa para atingir um objetivo específico. A Figura 3.7 decompõe o objetivo em sub-objetivos e operações, permitindo identificar o conhecimento requerido e os artefatos necessários no curso da tarefa. Elementos cruciais incluem o Gatilho (o que motiva a ação) e a Saída pretendida (como o usuário sabe que terminou com sucesso).

A ideação não deve ser um processo livre de regras, mas uma jornada estruturada que utiliza métodos diversos para garantir que a solução final seja viável, útil e empática.

3.7 Ferramentas de Prototipação e Validação de Ideias

O Design Sprint é um modelo focado na ideação e validação rápida de ideias em um período fixo de cinco dias. O objetivo é responder perguntas críticas de negócios através do design, prototipagem e teste com usuários, funcionando como um "atalho" para aprender sem a necessidade de construir e lançar um produto real imediatamente.

3.7.1 Design Sprint

A estrutura da semana de sprint é dividida da seguinte forma:

- *Primeiro dia*: Dedicado ao entendimento do problema, criação de um mapa de desafio e definição de metas de longo prazo.
- *Segundo dia*: Focado na inspiração e no esboço individual de soluções concorrentes no papel.
- *Terceiro dia*: Momento de crítica, onde a equipe analisa as propostas e escolhe as ideias que serão testadas.
- *Quarto dia*: Construção de um protótipo realista (simulando o produto) e criação de um *storyboard* para guiar o teste.
- *Quinto dia*: Realização de testes de usabilidade com usuários reais para obter aprendizado e clareza sobre os próximos passos.

3.7.2 Prototipação

A prototipação é definida como o processo de simular o resultado de um produto para realizar validações com clientes e embasar o desenvolvimento. A fidelidade de um protótipo é definida por quatro dimensões: grau de funcionalidade, similaridade de interação, detalhamento e refinamento estético.

Os níveis de fidelidade são classificados em:

- *Baixa fidelidade* (Rabiscoframes): Esboços rápidos em papel que focam na lógica e no conceito da interface, permitindo que a equipe explore alternativas sem compromisso com a estética.
- *Média fidelidade* (Wireframes): Telas geralmente em tons de cinza, focadas na hierarquia da informação, fluxo de navegação e blocos de estrutura.
- *Alta fidelidade*: Protótipos que aplicam a identidade visual, cores e componentes reais. São os mais próximos do produto real e ideais para testes de usabilidade e apresentações para stakeholders. Ferramentas como Figma, Adobe XD e Sketch são as mais comuns para esta etapa.

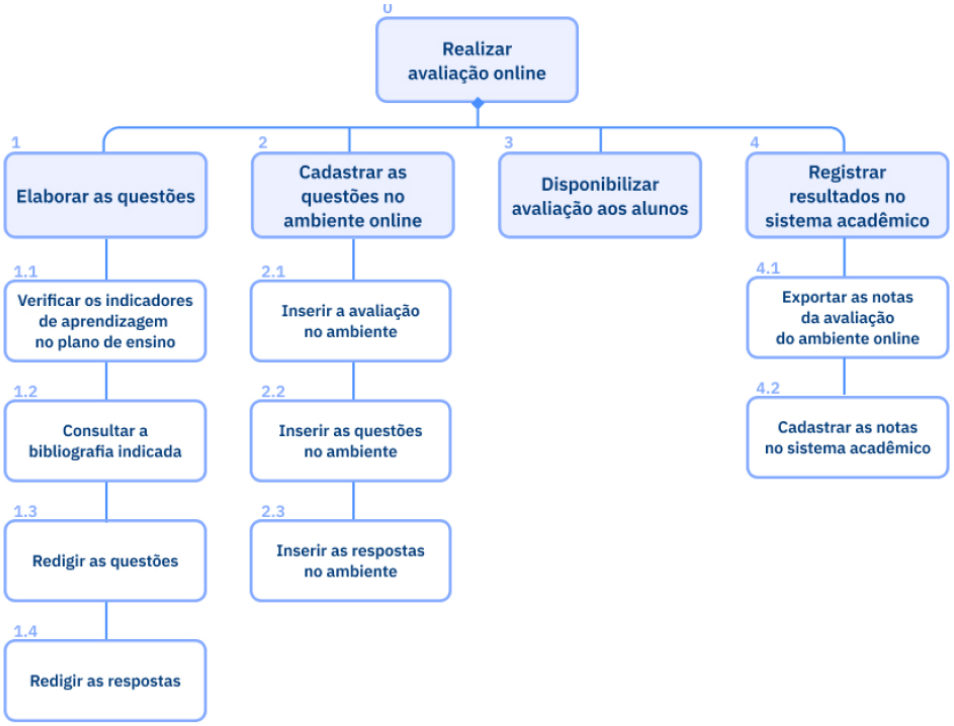

Figura 3.7: Hierarchical Task Analysis (HTA)

3.7.3 Design Studio

O Design Studio é uma ferramenta de criação coletiva que utiliza uma equipe multidisciplinar para gerar soluções através de ciclos de divergência e convergência. Baseado na metodologia de Brian K. Sullivan, o processo busca tirar o time da zona de conforto e identificar escopos de forma visual e convidativa. O método é executado em quatro etapas principais:

- Apresentação do problema, personas, requisitos e restrições.
- Geração de ideias individuais através de esboços e rodadas de feedback construtivo.
- Refinamento das propostas, onde o grupo se une para propor uma nova solução baseada nos pontos fortes das ideias anteriores.
- Priorização e seleção da solução que melhor satisfaz o problema inicial.

3.7.4 Termos de Uso

Esta seção destaca a importância do documento de Termos de Uso, que serve para informar o titular dos dados sobre o funcionamento do sistema e o tratamento de seus dados pessoais. Para estar em conformidade com a LGPD e os princípios do Governo Digital, este documento deve ser transparente e listar claramente os direitos e responsabilidades tanto dos usuários quanto do prestador de serviço. A construção deste termo deve ocorrer após alinhamentos com profissionais de proteção de dados e validação com o cliente.

3.7.5 Checklist de Privacidade

O Checklist de Privacidade é uma ferramenta composta por oito perguntas sequenciais que visam garantir que os aspectos de segurança e privacidade sejam atendidos desde a concepção (*Privacy by Design*). O checklist aborda questões como:

- Os dados pessoais tratados estão de acordo com a finalidade proposta?
- Há uma hipótese legal clara para o tratamento de dados? Qual?
- A maneira proposta de garantir os direitos do titular é a melhor forma para o módulo ou funcionalidade?
- Há regras de acesso aos dados pessoais?
- Há regras de compartilhamento?
- Foram compartilhados dados com terceiros? Se sim, quais dados e com quem/o que?
- Há necessidade de trabalhar com o consentimento enquanto padrão ético?
- Houve a atualização/elaboração dos termos de uso e políticas de privacidade?

A aplicação deste checklist na etapa de validação de propostas evita retrabalhos futuros e garante que o produto já nasça adequado às normas vigentes de proteção de dados

3.8 Ferramentas de Delimitação e Refinamento

Esta etapa foca no processo de definir com precisão o que será elaborado e refinar as soluções para que estas atendam aos requisitos reais dos usuários. Estas ferramentas atuam como uma ponte crucial entre as macro etapas de Descoberta (Discovery) e a construção efetiva na macro etapa de Entrega (Delivery).

Abaixo, os principais conceitos e ferramentas:

- *Canvas de Proposta de Valor* (*CPV*): Esta ferramenta é utilizada para garantir o alinhamento entre o que o produto oferece e o que o cliente realmente necessita. O CPV é dividido em dois blocos principais:
    - *Perfil do Cliente*: Onde são mapeadas as tarefas (*jobs*) do cliente, suas dores (*pains*) e os ganhos esperados (*gains*).
    - *Mapa de Valor*: Onde se descrevem os produtos e serviços, os aliviadores de dor e os criadores de ganho. O objetivo final é alcançar o "Encaixe" (Fit), que ocorre quando os itens do mapa de valor endereçam as necessidades identificadas no perfil do cliente.

- *Análise de Conteúdo* (*Content Audit*): Consiste em listar e relacionar todo o conteúdo para obter uma visão sistêmica do produto. Essa análise revela a profundidade das informações e ajuda a criar estratégias para que o sistema atue de forma conjunta e coerente.
- *Critérios de Aceitação*: São os requisitos mínimos necessários para que uma história de usuário seja aceita como pronta pelo time de desenvolvimento. Eles servem como parâmetros para que o Product Owner considere a funcionalidade concluída. A estrutura detalhada sugerida segue o modelo: "Dado que <*situação*>, quando <*ação*>, então <*resultado esperado*>".
- *Jornada do Usuário*: Mapeia todos os pontos de contato do usuário com o produto, permitindo a visualização de todo o percurso. Inclui o cenário, metas, ações, sentimentos e oportunidades identificadas em cada etapa.
- *Mapa de Empatia*: Proporciona uma compreensão profunda das dores, sentimentos, pensamentos e comportamentos do usuário. Ele responde a perguntas como "O que o usuário ouve?", "O que ele vê?" e "O que ele sente?".
- *User Flow*: Descreve o caminho percorrido pelo usuário para realizar uma atividade específica dentro do produto digital. Enquanto os wireframes focam no layout da tela, o User Flow foca na materialização do processo e das ações.
- *Prova de Conceito* (*PoC*): Utilizada para verificar a viabilidade técnica de uma ideia inovadora ou arriscada antes de um investimento massivo. Um documento de PoC deve identificar problemas de desempenho e detalhes da arquitetura do software.
- *Técnicas de Priorização*: Para gerenciar o escopo e manter o foco no valor, usam-se diversas técnicas como:
    - *MoSCoW*: Classifica demandas em *Must have* (essencial), *Should have* (importante), *Could have* (desejável) e *Won't have* (não terá neste momento).
    - *Matriz GUT*: Prioriza problemas e riscos com base na Gravidade, Urgência e Tendência.
    - *Buy-a-feature*: Dinâmica que simula a compra de funcionalidades com um orçamento fictício para validar prioridades sob a ótica de valor para o usuário.
    - *Matriz Impacto vs Esforço*: Auxilia a focar em funcionalidades de alto impacto e baixo esforço técnico.
- *Levantamento de Dados e LGPD*. Esta etapa define quais dados serão utilizados no sistema, respeitando os princípios da Lei Geral de Proteção de Dados (LGPD). É essencial organizar o fluxo de dados para estabelecer questões sobre coleta, armazenamento e eliminação das informações pessoais.

3.9 Ferramentas de Avaliação e Melhorias

Esta etapa foca nos métodos de validação de soluções e protótipos para garantir que o produto atenda às necessidades reais dos usuários e às estratégias de negócio

3.9.1 *Card Sorting* e *Survey*

O *Card Sorting* é uma técnica fundamental para compreender os modelos mentais dos usuários e organizar a arquitetura da informação. Os participantes recebem cartões com etiquetas do sistema e os agrupam conforme sua lógica. O método pode ser aberto (o usuário nomeia as categorias), fechado (categorias pré-definidas) ou híbrido. Recomenda-se entre quinze e vinte participantes para identificar os padrões mais comuns.

Já o *Survey* (pesquisa) é utilizado para coletar dados quantitativos e qualitativos sobre um grupo. Sua construção deve focar em perguntas claras, escalas validadas e evitar vieses para que o tratamento estatístico posterior gere indicadores confiáveis.

3.9.2 Teste A/B e Teste Alfa

O Teste A/B compara duas versões de uma mesma variável (como a cor de um botão ou o texto de um alerta) para identificar qual apresenta melhor desempenho. O processo envolve definir um objetivo claro, separar grupos de usuários de forma aleatória e medir o resultado de forma isolada.

O Teste Alfa é uma etapa de validação interna e exploratória realizada pela própria equipe de desenvolvimento (designers, analistas e desenvolvedores). O objetivo é encontrar inconsistências básicas e erros de lógica antes que o produto chegue ao usuário final.

3.9.3 Teste de Aceitação

Os testes de aceitação visam estabelecer a confiança de que o sistema satisfaz os requisitos documentados. O processo segue o ciclo PDCA (Planejar, Desenvolver, Executar e Contornar Resultados). Existem diferentes tipos, como o UAT (focado no usuário), OAT (operacional), CAT (contratual) e o RAT (regulamentar, focado em conformidade legal).

3.9.4 Teste de Usabilidade e Teste dos Cinco Segundos

O Teste de Usabilidade consiste em observar usuários reais realizando tarefas específicas no sistema para identificar falhas de interface e fluxo. Segundo Nielsen [2012], testar com apenas 5 usuários costuma ser suficiente para identificar a maioria dos problemas de usabilidade. As métricas coletadas incluem taxa de sucesso, tempo de execução e esforço percebido.

Complementarmente, o Teste dos Cinco Segundos mede a percepção imediata do usuário sobre a clareza da mensagem principal de uma tela após uma exposição curtíssima.

3.9.5 Validação com o Cliente e Ferramentas de Satisfação

A Validação de propostas com o cliente é um rito estratégico para alinhar expectativas, expor riscos e evitar retrabalhos. Para medir o sucesso de forma objetiva, o laboratório utiliza:

- *SUS* (*System Usability Scale*): Questionário de dez perguntas que gera uma pontuação de 0 a 100. Pontuações acima de 68 são consideradas aceitáveis.
- *NPS* (*Net Promoter Score*): Avalia a lealdade do usuário, classificando-os em Promotores, Neutros ou Detratores.

3.9.6 Métricas e Segurança

A gestão baseada em dados utiliza o conceito de OMTM (*One Metric That Matters*), focando em uma métrica central para guiar as estratégias. Indicadores comuns incluem Aquisição, Ativação, *Churn* (taxa de cancelamento) e Receita. Por fim, os Testes de Segurança garantem a proteção dos dados conforme o Art. 46 da LGPD. O procedimento operacional (POP) do Bridge envolve o refinamento de requisitos de segurança, modelagem de ameaças e execução de testes de API, injeção de código e quebra de autenticação.

Este capítulo reforça que o processo de software centrado no usuário é cíclico: a avaliação constante gera aprendizado que alimenta a melhoria contínua do produto.

3.10 Comentários Finais: Como transformar o processo da sua empresa em centrado no usuário?

Esta seção final aborda a transição de um foco apenas em produtos para uma mudança na cultura organizacional.

A implantação de um processo centrado no usuário não ocorre de forma instantânea; ela exige uma conscientização profunda sobre a importância estratégica da experiência do usuário (UX) para atingir os objetivos de negócio. Sugere-se uma sequência de cinco passos fundamentais para guiar essa transição:

- *Reconhecer a necessidade de mudança*: O ponto de partida é admitir que o mercado e os usuários estão mais exigentes e que a experiência do usuário se tornou um diferencial competitivo essencial.
- *Avaliar o estado atual*: Antes de evoluir, a organização deve realizar uma reflexão honesta sobre seus gargalos internos e aspectos negativos na relação com os usuários.
- *Identificar oportunidades de melhoria*: Utilizar dados concretos, como *logs* de erro, histórico de *bugs* e chamados de suporte, para mapear onde o design pode gerar impacto imediato.
- *Implantar* quick-wins (vitórias rápidas): Focar em iniciativas de UX de alto impacto e baixo esforço que tragam retornos positivos em curto prazo para gerar confiança no novo processo.
- *Melhorar continuamente*: Envolver todos os interessados na avaliação constante do produto, garantindo que o aprendizado se torne parte do DNA da empresa.

Baseando-se em Pernice *et al*. [2021], destacam-se quatro pilares que sustentam a cultura voltada ao usuário: Estratégia, Cultura, Processo e Resultados. A transformação deve garantir que os métodos de UX não sejam vistos apenas como ferramentas isoladas, mas como uma ponte entre a macro etapa de Descoberta e a de Entrega.

Para que uma organização saiba onde está e para onde deve ir, o capítulo detalha os seis níveis de maturidade em UX, adaptados do Nielsen Norman Group:

- *Nível 1 - Inexistente*: A UX é ignorada ou não reconhecida como importante. Não há esforços para entender as necessidades dos usuários.
- *Nível 2 - Limitado*: Práticas de UX ocorrem de forma esporádica em alguns projetos, mas não são sistemáticas nem culturais dentro da empresa.

- *Nível 3 - Emergente*: A organização reconhece o papel crítico da UX, mas a abordagem ainda é inconsistente e o planejamento de métodos é falho.
- *Nível 4 - Estruturado*: A UX é padronizada e as equipes colaboram de forma consistente em todos os processos de desenvolvimento.
- *Nível 5 - Integrado*: A UX está consolidada na estratégia de negócio. Valoriza-se a experimentação e a inovação como formas de aprendizado contínuo.
- *Nível 6 - Centrado no Usuário*: O estágio pleno, onde o pensamento focado no usuário guia toda a organização e a alta gestão mensura resultados a partir da maturidade dos processos.

A transformação para os níveis superiores exige um aumento na conscientização e aceitação da UX como um pilar de consecução de objetivos.

Este capítulo encerra enfatizando que o design centrado no usuário deve ser compreendido como uma visão transformadora da organização como um todo, integrando-se plenamente ao ciclo de vida de cada produto desenvolvido

REFERÊNCIAS

[Bono 2022] Bono, E. **Six Thinking Hats**. 1999.

[Brasil 2018] Brasil. Lei nº 13.709, de 14 ago. 2018. Lei Geral de Proteção de Dados Pessoais (LGPD). **Diário Oficial da União**, p. 59-59, 2018.

[Nielsen 2012] Nielsen, Jakob (2012). **How Many Test Users in a Usability Study?** Disponível em: https://www.nngroup.com/articles/how-many-test-users/ Acesso em 22 jun. 2026.

[Osborn 1979] Osborn, A. **Applied imagination:** principles and procedures of creative thinking. New York: Scribner, 1979.

[Pernice 2021] Pernice, K.; Gibbons, S.; Moran, K.; Whitenton, K. (2021) **The 6 Levels of UX Maturity**. Nielsen Norman Group. Disponível em: www. nngroup.com/articles/ux-maturity-model Acesso em: 21 ago. 2023.

[Wazlawick 2025] Wazlawick, R. S.; Santos, R. A.; Hammes, J. F.; Cunha, C. L. **Desenvolvimento de Software Centrado no Usuário: Construindo produtos com b_thinking, UX e agile**. 1. ed. Florianópolis: UFSC, 2025. v. 1. 343p.

Capítulo

4

Otimização de Prompts em Modelos de Linguagem em Larga Escala: Técnicas, Avaliação e Desafios

Nicollas R. de Oliveira (UFF), Dianne S. V. Medeiros (UFF),
Diogo M. F. Mattos (UFF)

Abstract

This chapter presents the main concepts, techniques, and challenges of prompt optimization in Large Language Models (LLMs), emphasizing their impact on the performance, control, and reliability of generated responses. It discusses approaches such as zero-shot, few-shot, chain-of-thought, and task decomposition, as well as more advanced techniques including prompt tuning and soft prompts. In addition, it explores evaluation methodologies based on metrics such as accuracy, consistency, and robustness, and analyzes limitations and risks, including vulnerabilities to prompt injection and generalization challenges. Finally, the chapter provides practical application examples and discusses research trends, encouraging a critical perspective on the efficient, safe, and reliable use of LLMs.

Resumo

Este capítulo apresenta os principais conceitos, técnicas e desafios relacionados à otimização de prompts *em modelos de linguagem de larga escala (*Large Language Models - *LLMs), enfatizando seu impacto no desempenho, no controle e na confiabilidade das respostas geradas. São discutidas abordagens como* zero-shot*,* few-shot*,* chain-of-thought *e decomposição de tarefas, bem como técnicas mais avançadas, incluindo* prompt tuning *e* soft prompts*. Ademais, são exploradas metodologias de avaliação baseadas em métricas como precisão, consistência e robustez, juntamente com a análise de limitações e riscos, como vulnerabilidades a* prompt injection *e desafios de generalização. Por fim, o capítulo apresenta exemplos práticos de aplicação e discute tendências de pesquisa, incentivando uma visão crítica sobre o uso eficiente, seguro e confiável de LLMs.*

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

4.1. Introdução

Os modelos de linguagem em larga escala (*Large Language Models* – LLMs) tornaram-se um dos principais componentes da atual geração de sistemas inteligentes, desempenhando papel central em agentes autônomos, assistentes conversacionais e sistemas de suporte à decisão em múltiplos domínios. Dados de relatórios recentes indicam que a taxa de adoção da inteligência artificial atingiu 88% das organizações globais, um incremento de 10% em relação ao ano anterior [Singla et al., 2025]. Esse avanço consolida a integração de arquiteturas baseadas em LLMs em sistemas de suporte à decisão estratégica, engenharia de software, gestão do conhecimento, ecossistemas de saúde e automação de atendimento ao cliente. Nesse contexto, o comportamento do modelo é fortemente influenciado pelo *prompt*, isto é, pelo conjunto de instruções, contexto e exemplos fornecidos como entrada. Diferentemente da programação tradicional, em que regras e fluxos de execução são explicitamente definidos, os LLMs dependem da interpretação semântica dessas instruções para produzir suas respostas [Du et al., 2026].

A importância dos *prompts* decorre da própria flexibilidade desses modelos, uma vez que uma mesma tarefa pode ser descrita por diferentes estruturas linguísticas, níveis de detalhamento e informações contextuais, produzindo múltiplas formas válidas de interação. Entretanto, pequenas alterações na formulação de um *prompt*, mudanças de contexto ou ambiguidades semânticas podem resultar em respostas significativamente distintas. Como consequência, aspectos relacionados à qualidade e confiabilidade das respostas tornam-se fatores críticos para o desenvolvimento de aplicações baseadas em LLMs. Essa elevada sensibilidade introduz desafios importantes relacionados à utilização prática desses modelos, destacando-se: **(i)** consistência, referente à capacidade do sistema de produzir respostas semelhantes para *prompts* semanticamente equivalentes; **(ii)** previsibilidade, associada à possibilidade de antecipar o comportamento do modelo diante de determinada formulação de instrução; e **(iii)** reprodutibilidade, relacionada à obtenção de resultados equivalentes quando o mesmo *prompt* é executado sob condições semelhantes.

Na prática, os desafios manifestam-se de diferentes formas [Campos et al., 2026]. Em muitos casos, instruções semanticamente próximas podem induzir comportamentos distintos, afetando diretamente a confiabilidade do sistema. Além disso, ambiguidades linguísticas, ausência de contexto adequado e formulações inadequadas podem aumentar a ocorrência de alucinações, vieses ou respostas inconsistentes. Além dos impactos sobre a qualidade das respostas, *prompts* inadequados também podem aumentar significativamente os custos operacionais associados ao uso de LLMs. Instruções excessivamente longas, ambíguas ou mal estruturadas tendem a consumir mais *tokens* durante a inferência e frequentemente exigem múltiplas interações até que o usuário obtenha o resultado desejado. Como consequência, aumentam-se o tempo de resposta, o consumo computacional e os custos financeiros associados à utilização de modelos comerciais, sobretudo em aplicações de larga escala e em domínios sensíveis, como saúde e finanças, nos quais erros podem estar associados a riscos concretos.

A crescente dependência da qualidade das instruções fornecidas transformou a construção de *prompts* em uma atividade central do desenvolvimento de aplicações baseadas em LLMs. Em muitos cenários, o desempenho observado depende menos de modificações na arquitetura do modelo e mais da forma como a tarefa é descrita. Inicialmente,

a construção de *prompts* era realizada predominantemente por tentativa e erro, baseada na experiência dos usuários e desenvolvedores. Entretanto, o aumento da complexidade das aplicações e a necessidade de resultados mais robustos e consistentes impulsionaram o surgimento de abordagens sistemáticas para criação, refinamento e avaliação de instruções. Esse movimento deu origem ao campo conhecido como Engenharia de *Prompt* (*Prompt Engineering*), posteriormente expandido para o conceito mais amplo de Otimização de *Prompt* (*Prompt Optimization*), que busca maximizar o desempenho dos modelos por meio da otimização explícita dos *prompts* utilizados durante a inferência [Saleem et al., 2026]. O objetivo não se restringe apenas à melhoria da qualidade das respostas, mas também à redução de ambiguidades, aumento da robustez, melhoria da eficiência operacional e adaptação do comportamento do modelo a tarefas específicas, incluindo a mitigação de riscos associados a ataques e incidentes de segurança [Moia et al., 2026].

Atualmente, a otimização de *prompts* engloba um conjunto diversificado de estratégias que vai desde abordagens baseadas em instruções e exemplos até técnicas avançadas de raciocínio e adaptação comportamental. Métodos como *zero-shot prompting*, *few-shot prompting*, *Chain-of-Thought*, *Tree-of-Thought*, *Program-of-Thought* e *Meta Prompting* demonstraram ganhos significativos em tarefas de classificação, perguntas e respostas, raciocínio lógico, programação e tomada de decisão. Paralelamente, abordagens automáticas passaram a empregar mecanismos de busca, aprendizado por reforço e até mesmo outros LLMs para gerar, avaliar e refinar *prompts* de forma iterativa, ampliando tanto o potencial de melhoria de desempenho quanto a superfície de ataque associada a *prompts* maliciosos. Além dos ganhos em desempenho, a otimização de *prompts* possui relevância prática crescente em diferentes domínios científicos e de engenharia, nos quais custos, riscos regulatórios e requisitos de auditabilidade se tornam fatores determinantes.

O restante do capítulo está organizado da seguinte forma. A Seção 4.2 introduz os fundamentos de *prompting*, definindo categorias de *prompts* e estratégias de condicionamento. A Seção 4.3 discute técnicas avançadas de otimização, incluindo abordagens paramétricas, métodos baseados em feedback humano e mecanismos de raciocínio estruturado. A Seção 4.4 apresenta *benchmarks*, critérios e métricas utilizados para avaliar *prompts* e modelos sob diferentes perspectivas. Em seguida, a Seção 4.4 analisa riscos e limitações associados à otimização de *prompts*, com ênfase em ameaças de segurança, *prompt leakage* e fenômenos como *overthinking*. Por fim, a Seção 4.6 sintetiza os principais pontos discutidos e aponta direções de pesquisa futuras em automação, robustez e integração da otimização de *prompts* em sistemas baseados em intenção.

4.2. Fundamentos de Prompting

Os *prompts* constituem o principal mecanismo de interação e controle em modelos de linguagem atuais, sendo compreendidos como uma instrução textual fornecida ao modelo com o objetivo de direcionar seu comportamento para execução de uma determinada tarefa. Essa abordagem permite explorar o conhecimento previamente aprendido pelos LLMs sem a necessidade de realizar novos processos extensivos de treinamento ou ajuste fino [Cui et al., 2025]. A efetividade de um *prompt* está diretamente relacionada à capacidade do modelo de interpretar corretamente o contexto, as instruções e os objetivos especificados pelo usuário. Dessa forma, fatores como clareza textual, escolha de palavras, organização estrutural, exemplos fornecidos e precisão semântica influenciam

significativamente a qualidade das respostas geradas.

Estruturalmente, um *prompt* é composto por elementos que fornecem contexto e orientações para a execução de uma tarefa específica. Dependendo da aplicação, esses elementos podem incluir instruções explícitas, informações contextuais, dados de entrada e restrições relacionadas ao formato esperado da resposta [Li et al., 2025a]. Em aplicações baseadas em *instruction tuning*, é comum a utilização do formato *instruction-input-output*, no qual o modelo recebe uma descrição da tarefa, os dados de entrada e uma especificação da saída desejada. Já sistemas de perguntas e respostas frequentemente utilizam estruturas do tipo *context-question-answer*, nas quais informações contextuais são apresentadas antes da consulta principal. Na prática, *prompts* em sistemas reais tendem a combinar esses elementos, por exemplo, definindo o papel do modelo, Geração Aumentada por Recuperação (*Retrieval-Augmented Generation* - RAG) e fornecendo um pequeno conjunto de exemplos antes da consulta principal.

De acordo com a forma como a informação é representada e utilizada durante o condicionamento do modelo, os *prompts* podem ser agrupados em diferentes categorias. Essas categorias diferem principalmente quanto ao grau de interpretabilidade humana, ao mecanismo de otimização empregado e à forma como influenciam o comportamento dos LLMs [Jain e Jindal, 2025]. Os *prompts* podem ser categorizados da seguinte forma:

- ***Hard Prompts* (ou *Textual Prompts*)**: correspondem à forma mais tradicional de interação com LLMs, sendo compostos explicitamente por palavras ou subpalavras pertencentes ao vocabulário do modelo e, consequentemente, totalmente interpretáveis por humanos. Nessa abordagem, a tarefa é especificada por meio de instruções escritas em linguagem natural, permitindo que usuários compreendam, modifiquem e validem diretamente o conteúdo fornecido ao modelo. Essa característica torna os *hard prompts* altamente flexíveis e amplamente aplicáveis em diferentes domínios. Entretanto, por dependerem exclusivamente da formulação textual, estão sujeitos a ambiguidades semânticas e elevada sensibilidade à escolha de palavras, à organização das instruções e ao contexto apresentado, fazendo com que pequenas modificações na redação possam produzir variações significativas no comportamento do modelo. Em aplicações práticas, *hard prompts* são frequentemente combinados com exemplos (*few-shot*) e instruções explícitas para reduzir essa sensibilidade.
- ***Soft Prompts***: utilizam representações contínuas treináveis, geralmente implementadas como vetores densos com a mesma dimensionalidade dos *embeddings*, representações vetoriais, processados pelo modelo. Diferentemente dos *hard prompts*, essas representações não possuem correspondência direta com palavras do vocabulário e, portanto, não são interpretáveis por humanos. Durante o treinamento, os vetores são ajustados para capturar padrões e informações relevantes para uma tarefa específica, atuando como mecanismos de condicionamento capazes de direcionar o comportamento do modelo sem a necessidade de modificar seus parâmetros internos. Essa característica proporciona elevada eficiência paramétrica e permite adaptar um mesmo modelo a diferentes aplicações por meio do treinamento de uma quantidade reduzida de parâmetros adicionais. Como limitações, destacam-se a necessidade de uma etapa de otimização específica e a dificuldade de interpretar o conhecimento armazenado nas representações contínuas aprendidas [Li et al., 2025a].

- ***Prompts Discretos Otimizados***: constituem uma extensão dos *hard prompts*, na qual os *tokens* que compõem a instrução são selecionados, modificados ou refinados automaticamente com o objetivo de maximizar o desempenho do modelo em uma determinada tarefa. Embora permaneçam integralmente representados no espaço textual e preservem sua interpretabilidade, a construção dessas instruções deixa de depender exclusivamente da elaboração manual realizada por especialistas. Em vez disso, algoritmos de otimização são empregados para explorar diferentes combinações de palavras, estruturas textuais ou padrões de instrução, buscando identificar formulações mais eficazes para direcionar o comportamento do modelo. Dessa forma, essa categoria procura combinar a transparência e a interpretabilidade dos *hard prompts* com a capacidade de adaptação proporcionada por mecanismos automáticos de otimização, aproximando-se de uma ponte entre *prompt engineering* manual e abordagens paramétricas.

As estratégias de *prompting* definem diferentes formas de estruturar instruções, exemplos e contexto com o objetivo de orientar o processo de inferência dos LLMs. Essas abordagens podem ser organizadas em diferentes categorias de acordo com o mecanismo utilizado para condicionar a geração das respostas. Uma das capacidades fundamentais que viabilizam diversas estratégias de *prompting* é o aprendizado em contexto (*In-Context Learning* – ICL). O ICL corresponde à capacidade dos LLMs de utilizar informações fornecidas diretamente no *prompt* para inferir padrões e produzir respostas compatíveis com a tarefa especificada, sem necessidade de atualização explícita de seus parâmetros internos. Em vez de aprender por meio de novos ciclos de treinamento, o modelo explora instruções, exemplos e informações contextuais presentes na entrada para ajustar sua inferência ao problema apresentado, o que torna a janela de contexto um recurso crítico tanto para desempenho quanto para custo e robustez.

4.2.1. Estratégias Baseadas em Exemplos

As estratégias baseadas em exemplos exploram a capacidade dos LLMs de utilizar demonstrações fornecidas diretamente no *prompt* para inferir padrões e orientar a geração das respostas. Essas abordagens diferem principalmente pela quantidade de exemplos disponibilizados ao modelo e pelo nível de informação fornecido para caracterizar a tarefa [Brown et al., 2020].

No ***zero-shot prompting***, o modelo recebe apenas uma descrição textual da tarefa, sem qualquer exemplo explícito de entrada e saída. Nesse cenário, o LLM deve inferir autonomamente o padrão esperado utilizando exclusivamente o conhecimento adquirido durante o pré-treinamento. Essa abordagem evidencia a capacidade de generalização semântica dos modelos, permitindo a execução de tarefas inéditas a partir de instruções em linguagem natural, ao custo de maior variabilidade de respostas.

O ***one-shot prompting*** estende esse paradigma ao incluir um único exemplo demonstrativo no *prompt*, que atua como referência estrutural para auxiliar o modelo na compreensão do formato, estilo ou raciocínio esperado para a tarefa. Já o ***few-shot prompting*** utiliza múltiplos exemplos demonstrativos antes da consulta principal, permitindo que o modelo identifique regularidades, padrões semânticos e relações contextuais mais complexas. Essa abordagem é amplamente empregada em tarefas que exigem adaptação

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

contextual, padronização de respostas ou aprendizado implícito de regras sem necessidade de *fine-tuning* adicional. Em contrapartida, o ***many-shot prompting*** representa uma extensão do paradigma *few-shot*, caracterizada pela utilização de uma quantidade significativamente maior de exemplos contextuais. Nessa estratégia, dezenas ou até centenas de exemplos podem ser incorporados ao *prompt*, limitados principalmente pela capacidade da janela de contexto do modelo. O aumento da quantidade de demonstrações tende a fornecer uma representação mais abrangente da tarefa, podendo melhorar desempenho, robustez e aderência ao formato desejado, especialmente em cenários complexos, especializados ou altamente dependentes de contexto [Li, 2023], mas também intensifica o custo computacional e o risco de vazamento de exemplos sensíveis.

4.2.2. Estratégias Baseadas em Instruções e Contexto

As estratégias baseadas em instruções e contexto concentram-se na definição explícita de objetivos, restrições operacionais e informações auxiliares utilizadas para orientar a geração das respostas. Embora possam ser combinadas com exemplos demonstrativos, essas abordagens enfatizam principalmente a especificação textual das informações necessárias para execução da tarefa. Nesse contexto, o modelo utiliza instruções, descrições de contexto e papéis semânticos para interpretar os requisitos da aplicação e adequar sua resposta ao cenário apresentado.

O ***instruction prompting*** utiliza instruções explícitas e detalhadas para especificar diretamente a tarefa que deve ser executada pelo modelo. Nesse paradigma, o objetivo da tarefa é descrito em linguagem natural, frequentemente incluindo restrições, critérios de execução, formato de saída e requisitos específicos relacionados à resposta desejada. Essa estratégia é amplamente empregada em sistemas conversacionais, geração de conteúdo, sumarização e automação de tarefas textuais devido à sua simplicidade e elevada interpretabilidade.

O ***contextual prompting*** expande essa abordagem ao incorporar informações contextuais adicionais relevantes para a tarefa. Esse contexto pode incluir documentos, descrições de domínio, regras operacionais, bases de conhecimento ou dados históricos capazes de complementar a compreensão do problema pelo modelo. O objetivo principal consiste em reduzir ambiguidades e aumentar a aderência das respostas ao cenário específico da aplicação, sendo particularmente útil em ambientes especializados e aplicações dependentes de conhecimento de domínio, mas exigindo cuidado com a integração de fontes potencialmente adversariais.

Por sua vez, o ***role prompting*** consiste na definição explícita de um papel, perfil ou especialização para o modelo durante a interação. Nessa estratégia, o LLM é instruído a assumir características associadas a determinados perfis profissionais, acadêmicos ou operacionais. A definição desse papel influencia diretamente o estilo linguístico, o nível de detalhamento técnico, a forma de argumentação e a estrutura das respostas produzidas, permitindo maior controle sobre o padrão de geração durante a inferência. Em cenários sensíveis, papéis mal definidos podem amplificar vieses ou facilitar contornos de salvaguardas, o que reforça a necessidade de projetar *prompts* com requisitos de segurança explícitos.

4.3. Técnicas Avançadas de Otimização

A otimização de *prompts* pode ser formulada como um problema de busca pela instrução que maximiza o desempenho de um modelo em uma determinada tarefa [Li, 2023]. Formalmente, esse objetivo pode ser representado pela Equação 1:

$$P^{\star} = \arg\max_{P} \mathbb{E}_{(x,y)\in D} \left[S(f_{\theta}(P,x),y) \right], \tag{1}$$

em que x representa a entrada, y a saída esperada pertencente ao conjunto de dados D, f_{θ} corresponde ao modelo de linguagem parametrizado por θ e S denota uma função de avaliação responsável por quantificar a qualidade da resposta produzida. Nesse contexto, o objetivo da otimização consiste em encontrar o *prompt* $P^{\star}$ capaz de maximizar o desempenho esperado do modelo segundo a métrica adotada.

Na prática, a busca por $P^{\star}$ enfrenta desafios relevantes. O espaço de *prompts* textuais é discreto, de alta dimensionalidade e não diferenciável, o que dificulta o uso direto de métodos de otimização baseados em gradiente. Além disso, a função de avaliação S é cara de computar, pois depende de chamadas ao modelo e pode ser ruidosa, especialmente quando o modelo é estocástico ou quando a avaliação depende de julgamentos humanos. Assim, técnicas de otimização de *prompts* tipicamente recorrem a heurísticas de busca, algoritmos evolutivos, métodos de reforço e aproximações de gradiente sobre representações contínuas, muitas vezes combinando esses mecanismos com *feedback* de modelos auxiliares ou de avaliadores humanos.

Embora estratégias baseadas em exemplos, instruções e contexto permitam direcionar a inferência dos LLMs sem modificar seus parâmetros internos, a qualidade das respostas continua fortemente dependente da capacidade dessas abordagens de representar adequadamente os requisitos da tarefa. Em cenários complexos ou especializados, essa limitação motivou o desenvolvimento de métodos capazes de adaptar ou otimizar os mecanismos de condicionamento utilizados pelos modelos. Nesse contexto, surgiram diferentes abordagens de otimização que atuam sobre distintos componentes do processo inferencial, incluindo a adaptação de representações associadas aos *prompts*, mecanismos automáticos de refinamento, estratégias baseadas em preferências e técnicas voltadas ao aprimoramento do raciocínio. As subseções seguintes apresentam os principais métodos utilizados para esse propósito.

4.3.1. Otimização Paramétrica

A otimização paramétrica compreende abordagens que adaptam LLMs para tarefas específicas por meio do treinamento de um conjunto reduzido de parâmetros adicionais, mantendo congelados os pesos do modelo original. Essas técnicas utilizam os *soft prompts*. Durante a otimização, apenas essas representações são atualizadas, resultando em métodos altamente eficientes em termos de memória e número de parâmetros treináveis. As principais abordagens dessa categoria diferenciam-se pela forma como os parâmetros adicionais são inseridos e propagados ao longo da arquitetura *Transformer*. Entre os métodos mais representativos destacam-se o *Prompt Tuning*, o *Prefix Tuning* e o *P-Tuning*.

O ***Prompt Tuning*** utiliza uma sequência de vetores treináveis concatenada aos *embeddings* da entrada antes do processamento pelo *Transformer* [Li et al., 2025b]. Es-

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

ses vetores possuem a mesma dimensionalidade dos *embeddings* do modelo, porém não correspondem a *tokens* reais do vocabulário. Durante o treinamento, apenas os parâmetros associados ao *prompt* são atualizados por retropropagação, enquanto os pesos do modelo permanecem inalterados. Como os parâmetros treináveis estão concentrados exclusivamente na camada de entrada, o método apresenta elevada eficiência paramétrica e requisitos de armazenamento reduzidos. Entretanto, a influência das representações aprendidas depende da propagação das informações através de todas as camadas da rede, limitando sua capacidade de modificar diretamente os estados internos produzidos pelo modelo em tarefas que exigem adaptações mais profundas.

O ***Prefix Tuning*** estende esse conceito de *Prompt Tuning* ao introduzir parâmetros treináveis diretamente nos mecanismos de atenção do *Transformer* por meio de prefixos, i.e. vetores contínuos que atuam como *prompts* virtuais acoplados às camadas do modelo. Em vez de atuar apenas sobre os *embeddings* de entrada, a abordagem aprende esses vetores embutindo-os diretamente nas estruturas de chave (*key*) e valor (*value*) utilizadas pelos blocos de atenção. Esses vetores funcionam como elementos contextuais persistentes, acessíveis durante todo o processamento da sequência. Como consequência, sua influência não se restringe à camada de entrada, permitindo atuar diretamente sobre as representações internas produzidas em múltiplos níveis da arquitetura. Os parâmetros aprendidos permanecem independentes da entrada processada e armazenam informações associadas à tarefa, possibilitando que diferentes aplicações compartilhem o mesmo modelo base utilizando conjuntos distintos de prefixos.

O ***P-Tuning*** foi proposto para ampliar a capacidade de representação dos *soft prompts*. Em sua formulação original, os vetores contínuos não são aprendidos de forma independente, mas gerados por uma rede neural auxiliar responsável por modelar dependências entre os diferentes *tokens* virtuais do *prompt*. Essa estratégia produz representações mais estruturadas e expressivas, permitindo capturar relações contextuais que dificilmente seriam representadas por *embeddings* otimizados de forma isolada. Posteriormente, o *P-Tuning v2* expandiu esse conceito ao distribuir parâmetros treináveis ao longo de múltiplas camadas do *Transformer*. Em vez de restringir o condicionamento à entrada do modelo, representações contínuas são inseridas em diferentes níveis da arquitetura, permitindo influenciar diretamente os estados internos produzidos durante a inferência. Essa modificação aumenta significativamente a capacidade de adaptação do método e amplia sua aplicabilidade para tarefas de compreensão, classificação, extração de informação e raciocínio. De forma geral, técnicas de otimização paramétrica ocupam uma posição intermediária entre *prompt optimization* puramente textual e *fine-tuning* completo, oferecendo um compromisso interessante entre custo de treinamento, capacidade de especialização e reuso de modelos base.

4.3.2. Aprendizado por Feedback Humano

O treinamento autoregressivo utilizado no pré-treinamento de LLMs permite que os modelos aprendam padrões estatísticos da linguagem, mas não garante que suas respostas estejam alinhadas com as preferências, expectativas ou objetivos dos usuários. Esse fenômeno, frequentemente denominado *objective mismatch*, decorre da diferença entre o objetivo de treinamento do modelo e os critérios utilizados por humanos para avaliar a qualidade de uma resposta, como utilidade, segurança, veracidade e adequação ao con-

texto [Liu, 2025]. Nesse cenário, o **Aprendizado por Reforço com *Feedback* Humano** (*Reinforcement Learning from Human Feedback* - RLHF) tornou-se uma das principais abordagens para incorporar preferências humanas diretamente ao processo de otimização dos modelos. O RLHF utiliza avaliações humanas para construir uma função de recompensa capaz de estimar a qualidade das respostas produzidas pelo modelo. Em vez de definir explicitamente regras ou métricas para cada tarefa, a abordagem assume que as preferências observadas nos avaliadores representam aproximações do comportamento desejado para o sistema. Dessa forma, o problema de alinhamento é reformulado como uma tarefa de maximização de recompensas, permitindo que técnicas de aprendizado por reforço sejam empregadas para direcionar a geração das respostas.

O processo normalmente inicia com uma etapa de *Supervised Fine-Tuning* (SFT), na qual o modelo é ajustado utilizando exemplos produzidos por humanos. Em seguida, múltiplas respostas geradas para uma mesma instrução são avaliadas e ranqueadas por anotadores. Essas preferências são utilizadas para treinar um modelo de recompensa (*Reward Model* – RM), responsável por estimar um valor escalar de recompensa associado ao grau de alinhamento de uma resposta com os critérios definidos pelos avaliadores. Em aplicações modernas, o treinamento do RM é frequentemente realizado a partir de comparações pareadas de respostas, permitindo que o sistema aprenda preferências relativas em vez de depender de avaliações absolutas. Após o treinamento do modelo de recompensa, o LLM passa a ser tratado como uma política de decisão que deve maximizar as recompensas atribuídas pelo RM. Nessa etapa, algoritmos de aprendizado por reforço, particularmente a *Proximal Policy Optimization* (PPO), ajustam os parâmetros do modelo para aumentar a probabilidade de gerar respostas consideradas preferíveis pelos avaliadores humanos. Como consequência, o modelo passa a produzir respostas mais alinhadas aos critérios desejados, mesmo que tais critérios não tenham sido explicitamente codificados durante o pré-treinamento [Sharma et al., 2025].

A principal contribuição do RLHF consiste em transformar preferências humanas, originalmente difíceis de formalizar matematicamente, em sinais de otimização capazes de orientar o treinamento do modelo. Entre suas principais vantagens destacam-se a capacidade de incorporar critérios subjetivos de avaliação, a melhoria da utilidade prática das respostas e a possibilidade de adaptar modelos para diferentes objetivos sem necessidade de especificar manualmente funções de recompensa [Chaudhari et al., 2025]. Ao mesmo tempo, o RLHF altera a função efetiva que *prompts* otimizados buscam explorar: instruções que eram eficazes em modelos não alinhados podem tornar-se menos eficientes após o alinhamento, e *prompts* adversariais passam a explorar fragilidades do modelo de recompensa. Assim, técnicas mais recentes, como *Direct Preference Optimization* (DPO) e *Reinforcement Learning from AI Feedback* (RLAIF), podem ser vistas não apenas como mecanismos de alinhamento, mas também como formas de regular o espaço de *prompts* eficazes e robustos, impactando diretamente a prática de otimização de *prompts*.

4.3.3. Raciocínio Estruturado

Embora os LLMs apresentem capacidades emergentes de raciocínio adquiridas durante o pré-treinamento, a geração direta da resposta nem sempre conduz à melhor solução para problemas que exigem múltiplas etapas de análise. Em tarefas envolvendo lógica, matemática, planejamento ou tomada de decisão, erros produzidos durante o pro-

cesso inferencial podem comprometer significativamente a resposta final. Para mitigar essa limitação, diversas abordagens passaram a explorar mecanismos explícitos de decomposição da inferência, nos quais o modelo produz representações intermediárias antes de emitir a solução final. Essas estratégias diferem principalmente pela forma como exploram, avaliam e selecionam possíveis trajetórias inferenciais durante a geração. Enquanto algumas abordagens utilizam uma única sequência de inferências, outras exploram múltiplas alternativas, realizam processos de agregação de respostas ou incorporam mecanismos externos de computação simbólica. Em comum, todas buscam aumentar a precisão, interpretabilidade e confiabilidade das respostas produzidas pelos LLMs.

O ***Chain-of-Thought*** (CoT) introduz explicitamente estados intermediários no processo de geração autoregressiva. Em vez de produzir diretamente a resposta final, o modelo gera uma sequência estruturada de inferências cuja saída parcial passa a compor o contexto utilizado para predição dos *tokens* subsequentes. Sob a perspectiva do *Transformer*, cada etapa adiciona novas evidências ao contexto disponível para os mecanismos de atenção, permitindo que decisões posteriores sejam condicionadas às conclusões previamente estabelecidas. Esse mecanismo reduz a complexidade de problemas que exigem múltiplas operações lógicas ou matemáticas ao decompor a tarefa em subtarefas menores e mais facilmente modeláveis. O CoT pode ser induzido por exemplos demonstrativos ou por instruções explícitas que incentivam a decomposição do processo inferencial em múltiplas etapas. Entretanto, como apenas uma trajetória é explorada, erros produzidos em etapas iniciais tendem a propagar-se ao longo de toda a cadeia de geração.

A abordagem ***Self-Consistency*** foi proposta para reduzir a dependência de uma única trajetória inferencial. Em vez de utilizar decodificação determinística, múltiplas cadeias de inferência são geradas por amostragem estocástica, produzindo diferentes caminhos para a resolução de um mesmo problema. Após a geração, as respostas finais são agregadas e a solução mais frequente é selecionada. O princípio subjacente consiste na hipótese de que trajetórias corretas tendem a convergir para uma mesma resposta, enquanto erros ocasionais apresentam maior dispersão. Sob essa perspectiva, o método realiza uma aproximação por amostragem do espaço de raciocínios possíveis, permitindo reduzir a influência de inferências inconsistentes sem necessidade de treinamento adicional. Como consequência, a abordagem frequentemente apresenta ganhos de desempenho em tarefas que exigem raciocínio matemático e dedução lógica complexa.

O ***Tree-of-Thought*** (ToT) generaliza o paradigma introduzido pelo CoT ao representar o processo inferencial como uma estrutura de busca em árvore. Em vez de gerar uma única sequência linear, o modelo produz múltiplos estados candidatos em cada etapa, formando ramos alternativos que podem ser expandidos, avaliados, podados ou revisitados ao longo da inferência. Cada estado corresponde a uma hipótese parcial para a solução do problema, permitindo que diferentes estratégias sejam exploradas simultaneamente. Mecanismos de busca em largura, busca em profundidade ou funções heurísticas podem ser empregados para selecionar quais ramos devem continuar sendo explorados. Diferentemente do CoT, que se compromete imediatamente com uma única sequência de inferências, o ToT possibilita a comparação entre soluções concorrentes e a realização de retrocessos quando caminhos pouco promissores são identificados. Essa característica aproxima o processo inferencial de técnicas clássicas de busca utilizadas em inteligência artificial, tornando a abordagem particularmente eficaz em problemas de planejamento,

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

otimização e tomada de decisão. Como contrapartida, a exploração simultânea de múltiplos ramos aumenta significativamente o custo computacional da inferência, elevando a demanda por *tokens* e tornando *prompts* de raciocínio estruturado especialmente sensíveis a custos e a fenômenos como *overthinking*.

Enquanto as abordagens anteriores operam exclusivamente no espaço textual, o ***Program-of-Thought*** (PoT) combina inferência em linguagem natural com execução simbólica. Nessa estratégia, o modelo gera expressões formais ou trechos de código que são executados por interpretadores externos, produzindo resultados posteriormente reincorporados ao contexto da geração. Em vez de depender exclusivamente dos parâmetros do modelo para realizar cálculos ou manipular estruturas lógicas complexas, parte do processamento é delegada a mecanismos determinísticos especializados. O processo inferencial passa então a alternar entre etapas de geração textual e execução computacional, permitindo combinar a flexibilidade dos LLMs com a precisão de ambientes formais de execução. Esse paradigma tem demonstrado elevada eficácia em tarefas envolvendo matemática, manipulação de tabelas, análise quantitativa e resolução de problemas algorítmicos, reduzindo significativamente erros de cálculo e inconsistências lógicas frequentemente observadas em abordagens puramente textuais [Saleem et al., 2026]. Em termos de otimização de *prompts*, técnicas de raciocínio estruturado ampliam a superfície de controle sobre o comportamento do modelo, mas também introduzem novos desafios relacionados a custo, calibração e segurança, como discutido nas seções seguintes.

4.4. Avaliação de *Prompts* e Métricas

A avaliação de estratégias de *prompting* e de otimização de *prompts* é normalmente realizada por meio de testes de desempenho (*benchmarks*) padronizados para avaliação de tarefas específicas. Cada *benchmark* é composto por um ou mais conjuntos de dados, métricas e procedimentos experimentais utilizados para mensurar o desempenho dos modelos em diferentes capacidades linguísticas e cognitivas. Como distintas estratégias de otimização podem impactar diferentes habilidades dos modelos, a literatura emprega *benchmarks* distribuídos em múltiplas categorias de tarefas, abrangendo desde classificação textual e inferência semântica até geração de conteúdo, extração de informação e raciocínio complexo. A diversidade dessas tarefas é particularmente importante no contexto da otimização de *prompts*, pois permite analisar como diferentes formas de instrução influenciam capacidades específicas e identificar cenários nos quais determinadas estratégias produzem maiores ganhos de desempenho ou geram efeitos adversos relevantes.

4.4.1. *Benchmarks* e Conjuntos de Dados

A **Inferência em Linguagem Natural** (*Natural Language Inference* – NLI) investiga a habilidade dos modelos de identificar relações lógicas entre uma premissa e uma hipótese. O *benchmark* **MNLI** (*Multi-Genre Natural Language Inference*) constitui uma das principais referências da área, contendo aproximadamente 433 mil pares de sentenças provenientes de múltiplos domínios e classificados como implicação, contradição ou neutralidade. Outros conjuntos amplamente utilizados incluem o **RTE** (*Recognizing Textual Entailment*), composto por cerca de 2,5 mil exemplos com elevado grau de dificuldade semântica, e o **CB** (*CommitmentBank*), voltado à interpretação de crenças, opiniões e es-

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

tados mentais. Como essas tarefas exigem integração contextual e raciocínio lógico, elas são particularmente adequadas para avaliar *prompts* que exploram instruções detalhadas, exemplos de raciocínio e estratégias como CoT, permitindo observar não apenas acurácia, mas também a sensibilidade do modelo a variações sutis na formulação da hipótese.

No contexto de **Perguntas e Respostas**, *Question Answering* (QA), os *benchmarks* avaliam a capacidade dos modelos de localizar, sintetizar ou inferir informações necessárias para responder perguntas. O **SQuAD** (*Stanford Question Answering Dataset*) tornou-se uma das principais referências da área ao disponibilizar mais de 100 mil perguntas associadas a artigos da Wikipédia. Enquanto o SQuAD 1.1 exige que a resposta esteja explicitamente presente no texto, o SQuAD 2.0 introduz aproximadamente 50 mil exemplos sem resposta válida, exigindo que o modelo reconheça situações em que a informação não está disponível. Outros conjuntos de dados amplamente empregados incluem o **BoolQ**, contendo cerca de 16 mil perguntas binárias derivadas de consultas reais de usuários, e o **CommonsenseQA**, formado por aproximadamente 12 mil questões de múltipla escolha construídas a partir da base ConceptNet. Nessa categoria, diferentes estratégias de *prompting* (por exemplo, *few-shot*, instruções explícitas sobre quando responder "não sei" ou *prompts* de verificação) permitem estudar como a forma de instrução afeta a propensão a alucinações e a capacidade do modelo de recusar respostas quando o contexto é insuficiente.

Em relação à **Geração de Texto**, a avaliação de tarefas generativas concentra-se na produção de textos coerentes, informativos e semanticamente consistentes. Em sumarização automática, o *benchmark* **CNN/DailyMail** disponibiliza aproximadamente 300 mil pares documento-resumo extraídos de notícias jornalísticas, exigindo a identificação e condensação de múltiplas informações relevantes. O **XSUM** complementa essa avaliação por meio de cerca de 227 mil exemplos focados em resumos altamente abstrativos, frequentemente representados por uma única sentença. Para tradução automática, os conjuntos associados às competições **WMT** (*Workshop on Machine Translation*) constituem a principal referência da área, abrangendo milhões de pares paralelos de sentenças em diferentes idiomas. Nessa categoria, a qualidade dos *prompts* costuma refletir-se diretamente na fidelidade semântica, na aderência às instruções e na qualidade linguística dos textos produzidos, permitindo comparar, por exemplo, *prompts* que enfatizam concisão, literalidade, criatividade ou preservação de termos técnicos.

As tarefas de **Similaridade Semântica** analisam a capacidade dos modelos de reconhecer equivalência ou proximidade de significado entre diferentes formulações linguísticas. O **STS-B** (*Semantic Textual Similarity Benchmark*) contém aproximadamente 8,6 mil pares de sentenças anotados por humanos com valores contínuos variando entre 0 e 5. O **QQP** (*Quora Question Pairs*) disponibiliza mais de 400 mil pares de perguntas rotuladas quanto à equivalência semântica, enquanto o **MRPC** (*Microsoft Research Paraphrase Corpus*) reúne cerca de 5,8 mil pares de sentenças provenientes de notícias jornalísticas classificados como paráfrases ou não paráfrases. Como a tarefa exige a identificação de significados equivalentes expressos por construções linguísticas distintas, esses conjuntos de dados são frequentemente utilizados para analisar a qualidade das representações semânticas produzidas pelos modelos sob diferentes *prompts*, avaliando, por exemplo, se *prompts* que solicitam explicações intermediárias favorecem ou prejudicam a capacidade de capturar nuances semânticas.

Uma categoria importante de *benchmarks* envolve a transformação de texto não estruturado em representações estruturadas, isto é, a capacidade de executar a tarefa de **Extração de Informação**. O **CoNLL03** é amplamente utilizado para reconhecimento de entidades nomeadas, contendo aproximadamente 22 mil sentenças anotadas com entidades dos tipos Pessoa, Organização, Localização e Miscelânea. O **CoNLL04** expande esse cenário ao incorporar também a extração de relações entre entidades presentes nos documentos. Já o **FewRel** disponibiliza cerca de 100 mil exemplos distribuídos entre 100 tipos de relações semânticas extraídas da Wikipédia, tendo sido projetado para cenários de aprendizado com poucos exemplos. Em aplicações práticas, essas tarefas aproximam-se diretamente de problemas de mineração de informação, construção automática de bases de conhecimento e descoberta de relações semânticas em grandes volumes de texto, sendo fortemente influenciadas pelo formato de saída especificado no *prompt* (por exemplo, extração em JSON, tabelas ou texto natural estruturado).

Os *benchmarks* de **Raciocínio** ocupam atualmente posição central na avaliação de estratégias avançadas de *prompting*. O **GSM8K** reúne aproximadamente 8,5 mil problemas matemáticos elaborados por professores e tornou-se uma das principais referências para avaliação de métodos de decomposição inferencial. Os conjuntos de dados **MultiArith** e **SVAMP** exploram problemas aritméticos de múltiplas etapas, sendo que o SVAMP introduz reformulações linguísticas destinadas a reduzir a exploração de padrões superficiais. O **AQUA-RAT** contém cerca de 100 mil questões de múltipla escolha acompanhadas de justificativas detalhadas, permitindo analisar simultaneamente a resposta final e o processo utilizado para obtê-la. Complementando esse cenário, o **BBH** (*BIG-Bench Hard*) reúne 23 tarefas selecionadas do *BIG-Bench* original por apresentarem elevada dificuldade para LLMs, abrangendo raciocínio lógico, planejamento, manipulação simbólica e conhecimento de senso comum. Esses conjuntos são especialmente úteis para comparar *prompts* com e sem CoT, ToT ou PoT, bem como para estudar fenômenos como *overthinking* e a relação entre comprimento da cadeia de raciocínio, custo e acurácia.

4.4.2. Critérios e Métricas de Avaliação

A avaliação da capacidade cognitiva de modelos de linguagem constitui um desafio complexo e abrangente, estimulando a criação de métricas automáticas que vão além da simples comparação entre textos. Tradicionalmente, a aferição da qualidade de respostas em PLN tem se baseado extensivamente na análise humana com anotadores. Embora esse método forneça referências de alta confiabilidade para a verdade fundamental (*ground truth*), ele apresenta limitações importantes, como a subjetividade das análises, o elevado tempo necessário para execução e a dificuldade de aplicação em larga escala [Chang et al., 2024]. Ademais, a complexidade crescente das saídas de LLMs e a vasta gama de cenários de aplicação tornam a avaliação manual impraticável e suscetível a vieses humanos e inconsistências nos julgamentos. Assim, o uso de uma estrutura avaliativa automatizada é fundamental. Para tal, são usados critérios de similaridade léxica, calibração e robustez, permitindo uma análise mais abrangente, eficiente e objetiva da capacidade de raciocínio, coerência e adaptabilidade dos LLMs sob diferentes *prompts*.

A **similaridade léxica** avalia a correspondência direta de vocabulário e a estrutura frasal, servindo como um indicador da fidelidade ou da relevância do conteúdo gerado em relação a uma base esperada. Essa avaliação é feita quantificando o grau de sobreposição

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

de termos e sequências de palavras entre o texto gerado por um modelo e um ou mais textos de referência. Para isso, podem ser empregadas diferentes métricas, classificadas em medidas de saída binária ou de saída contínua [Chang et al., 2024, Yang et al., 2024]. Em avaliações de otimização de *prompts*, métricas lexicais são úteis para comparar rapidamente diferentes formulações, mas não capturam plenamente mudanças no raciocínio interno do modelo ou na qualidade da justificação gerada.

Entre as métricas binárias, destacam-se *Exact Match* (EM) e *Quasi-Exact Match*. A métrica EM é uma medida binária utilizada para avaliar o grau de correspondência exata entre a saída produzida por um modelo e a resposta de referência em tarefas de geração de texto. Em cenários de QA, o EM assume valor 1 se o texto gerado for idêntico à resposta esperada, e 0 caso contrário. A QEM estende a EM ao flexibilizar a exigência de equivalência perfeita entre os textos comparados. A QEM admite pequenas variações textuais ao comparar a resposta gerada com a referência, por meio da aplicação de uma etapa de pré-processamento, que inclui a normalização para minúsculas, a remoção de espaços em branco e pontuações excessivas, além da exclusão de artigos definidos ou indefinidos. Assim, diferenças superficiais de formatação e de palavras com baixa carga semântica não têm alto impacto no significado semântico essencial da resposta, de forma que respostas semanticamente equivalentes sejam consideradas corretas mesmo quando não apresentam correspondência textual exata. Consequentemente, a QEM oferece uma medida mais robusta e menos sensível a variações estilísticas, o que é particularmente relevante em cenários nos quais *prompts* diferentes produzem saídas semanticamente idênticas, mas estruturalmente distintas.

Entre as métricas de saída contínua mais utilizadas, o *F1-Score* e o *ROUGE Score* destacam-se por fornecerem medidas mais graduais da qualidade das respostas. Enquanto o *F1-Score* combina precisão e revocação em uma média harmônica, avaliando simultaneamente a proporção de termos relevantes recuperados e a exatidão desses termos em relação ao texto de referência, o *ROUGE Score* mensura a similaridade entre textos por meio da sobreposição de unidades linguísticas, sendo amplamente empregado em tarefas de sumarização automática e geração de texto. Outra métrica contínua amplamente utilizada é a **BLEU** (*Bilingual Evaluation Understudy*). Essa métrica avalia a qualidade de textos gerados por máquinas, especialmente em tarefas como tradução automática e sumarização de texto, comparando o texto gerado com um ou mais textos de referência e quantificando a similaridade com base na sobreposição de n-grams, *i.e.* sequências de palavras. A métrica é calculada por

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

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

METEOR (*Metric for Evaluation of Translation with Explicit ORdering*) é uma

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

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

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

em que $F_{\text{média}}$ é a média harmônica ponderada da precisão e do *recall*, calculada como $F_{\text{média}} = \frac{10 \cdot \text{Precisão} \cdot \text{Recall}}{9 \cdot \text{Precisão} + \text{Recall}}$. O termo P é uma penalidade que varia entre $[0, 1]$, projetada para ajustar o *score* de acordo com a desordem na sequência das palavras. Essa penalidade é calculada com base no número de correspondências e de blocos de palavras não consecutivas entre a tradução gerada e a referência, penalizando traduções com ordem de palavras significativamente diferente da esperada. Em avaliações de *prompts*, métricas como BLEU e METEOR são úteis para comparar estilos de instrução em tradução e sumarização, mas podem atribuir *scores* altos a respostas fluentemente erradas em termos de conteúdo factual.

Além de métricas baseadas em sobreposição léxica, como BLEU e ROUGE, o **MoverScore**[1] [Zhao et al., 2019] e o **BERTScore**[2] [Zhang et al., 2020] representam métricas avançadas que utilizam *embeddings* contextuais para avaliar a similaridade semântica entre textos gerados por LLMs e suas referências. Ambas iniciam com a geração de *embeddings* para cada palavra dos textos candidato e de referência, geralmente empregando modelos pré-treinados como o BERT. O MoverScore então se diferencia ao calcular distâncias ponderadas entre pares de *embeddings* e resolver um problema de transporte ótimo (*Earth Mover's Distance* – EMD). Esse problema busca o custo mínimo para mover a massa semântica das palavras do texto candidato para a distribuição de palavras do texto de referência, quantificando o esforço necessário para que os significados do texto gerado se alinhem com os da referência. Em contraste, o BERTScore calcula a similaridade de cosseno entre cada *token* do candidato e seu *token* mais similar na referência, agregando esses *scores* em métricas de precisão, *recall* e *F1-score*. Enquanto o MoverScore se concentra no custo de transporte semântico global, o BERTScore avalia a similaridade contextual de *tokens* individuais e sua agregação. Do ponto de vista de otimização de *prompts*, métricas semânticas são particularmente úteis para comparar *prompts* que induzem justificativas alternativas, mas, assim como as métricas lexicais, podem atribuir *scores* altos a respostas plausíveis porém factualmente incorretas, o que reforça a necessidade de complementar essas métricas com avaliações focadas em veracidade e raciocínio.

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

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

lidade do modelo, especialmente em aplicações críticas. Em avaliações de *prompts*, a calibração torna-se relevante porque diferentes formulações podem induzir o modelo a emitir respostas mais ou menos confiantes, mesmo quando a acurácia permanece constante.

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

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

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

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

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

Em contextos de otimização de *prompts*, métricas de calibração permitem comparar formulações que explicitamente solicitam estimativas de confiança ou justificativas, verificando se essas instruções melhoram a calibração percebida das respostas.

O critério de **robustez** avalia a capacidade de um modelo manter seu desempenho e funcionalidade sob condições de entrada desafiadoras ou perturbadas. Tal avaliação engloba a resiliência do modelo a ataques adversariais, eventuais mudanças na distribuição dos dados e a presença de ruído ou variações na entrada. Sendo assim, a robustez é um critério fundamental para garantir a confiabilidade e a segurança dos modelos em ambientes do mundo real, nos quais os dados de entrada podem não ser idênticos aos dados de treinamento ou podem ser manipulados intencionalmente. Dentre as métricas fundamentais baseadas nesse critério, destacam-se a taxa de sucesso de ataque e a taxa de queda de desempenho, que capturam diretamente o impacto de *prompts* adversariais sobre modelos otimizados.

A **Taxa de Sucesso de Ataque** (*Attack Success Rate* – ASR) serve como uma métrica primordial para quantificar a robustez adversarial em LLMs [Wang et al., 2023]. Especificamente, para um conjunto de dados de validação $D = \{(x_i, y_i)\}_{i=1}^{N}$, onde x_i é a amostra de entrada e y_i é sua verdade fundamental (*ground truth*) correspondente, e dado

um método de ataque adversarial A, o ASR mede a proporção de ataques bem-sucedidos. Um ataque é considerado bem-sucedido quando, para uma entrada original x que o modelo substituto f classifica corretamente ($f(x) = y$), o exemplo adversarial gerado $A(x)$ induz o modelo a produzir uma predição incorreta ($f(A(x)) \neq y$). A taxa de sucesso é calculada como

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

em que $I[\cdot]$ é a função indicadora, que retorna 1 se a condição dentro dos colchetes for verdadeira e 0 caso contrário. Em estudos de otimização de *prompts*, o ASR quantifica quão facilmente uma estratégia de ataque pode explorar *prompts* aparentemente benignos para induzir comportamentos indevidos, mesmo em modelos alinhados.

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

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

em que A representa a função de ataque adversarial aplicada ao *prompt*, P denota a concatenação do *prompt* com a entrada x e M é a função de avaliação de desempenho que varia de acordo com a tarefa específica. Uma PDR elevada indica uma baixa robustez do modelo a alterações no *prompt*, o que é particularmente preocupante em sistemas que dependem de *prompts* compartilhados ou gerados automaticamente em ambientes potencialmente hostis.

4.5. Riscos e Limitações da Otimização de *Prompt*

Embora a otimização de *prompts* tenha se consolidado como um dos principais mecanismos para adaptação e controle de modelos de linguagem, sua utilização introduz desafios relacionados à segurança, robustez, privacidade e confiabilidade das respostas geradas. Uma vez que o funcionamento dos LLMs depende fortemente das instruções textuais fornecidas durante a inferência, vulnerabilidades associadas à manipulação do contexto, ambiguidades linguísticas e exposição de informações sensíveis tornam-se potenciais vetores de ataque ou degradação de desempenho [Sandoval et al., 2023, Moia et al., 2026]. Paralelamente, limitações inerentes ao processo de otimização podem comprometer a capacidade de generalização dos *prompts*, aumentar custos computacionais e reduzir a previsibilidade dos resultados.

4.5.1. Ameaças e Vulnerabilidades

Grande parte das ameaças e vulnerabilidades observadas em sistemas baseados em LLMs decorre da forma como o contexto é processado durante a inferência. Como instruções, exemplos e dados externos são representados e analisados conjuntamente pelos mecanismos de atenção, conteúdos inseridos na janela de contexto podem influenciar diretamente o comportamento do modelo, criando oportunidades para manipulação adversarial, vazamento de informações e contorno de mecanismos de alinhamento.

Nesse cenário, os ataques de **injeção de prompt** (*prompt injection*) constituem uma das ameaças mais relevantes à segurança de aplicações baseadas em LLMs. Nesses ataques, instruções inseridas por usuários ou provenientes de fontes externas são projetadas para modificar o comportamento originalmente esperado do modelo, induzindo-o a ignorar restrições de segurança, alterar objetivos previamente estabelecidos ou executar ações não autorizadas. O problema torna-se particularmente crítico em arquiteturas baseadas em *Retrieval-Augmented Generation* (RAG), agentes autônomos e sistemas integrados a ferramentas externas, uma vez que documentos recuperados dinamicamente passam a compor o contexto utilizado durante a inferência.

Uma consequência direta dessa limitação é o fenômeno de ***jailbreaking***, no qual o atacante procura contornar mecanismos de alinhamento e salvaguardas incorporados ao modelo. Diferentemente de ataques tradicionais que exploram vulnerabilidades de implementação, essa classe de ataque explora a própria capacidade semântica dos LLMs de reinterpretar instruções e restrições contextuais. Estratégias baseadas em manipulação de personas, cenários hipotéticos, reformulações linguísticas, encadeamento de instruções e raciocínios condicionais induzem o modelo a produzir conteúdos que deveriam ser bloqueados pelos mecanismos de alinhamento. Embora técnicas como RLHF reduzam significativamente a probabilidade de respostas inadequadas, trabalhos recentes mostram que tais mecanismos permanecem suscetíveis a entradas adversariais cuidadosamente construídas, evidenciando a ausência de garantias formais de segurança durante a inferência.

A evolução dos ataques adversariais também transformou a construção de *prompts* maliciosos em um problema de otimização. Inicialmente desenvolvidos por tentativa e erro, esses ataques passaram a utilizar métodos baseados em gradientes para explorar o espaço de *embeddings* e identificar sequências de *tokens* capazes de maximizar a probabilidade de respostas específicas ou contornar mecanismos de proteção. Estudos recentes descrevem ainda o emprego de algoritmos genéticos, estratégias de reflexão, arquiteturas multiagente e processos iterativos de busca para gerar ataques semanticamente coerentes, transferíveis entre modelos distintos e eficazes mesmo em cenários de caixa-preta, ampliando significativamente sua capacidade de automação e escalabilidade [Zhang et al., 2025].

Outra ameaça relevante corresponde ao **vazamento de informações** (*prompt leakage*). Essa ameaça explora vulnerabilidades como o compartilhamento da mesma janela de contexto por instruções de sistema, exemplos demonstrativos, documentos recuperados e dados do usuário e implicam que respostas geradas podem revelar parcial ou integralmente conteúdos que deveriam permanecer restritos. Esse problema inclui a exposição de *system prompts*, regras de negócio, documentos utilizados em mecanismos RAG, credenciais, informações corporativas e dados pessoais presentes durante a inferência. Além dos impactos relacionados à privacidade e à confidencialidade, a obtenção dessas informações pode facilitar a elaboração de novas tentativas de *prompt injection* e *jailbreaking*, ampliando progressivamente a superfície de ataque do sistema [Sandoval et al., 2023].

4.5.2. Limitações

Apesar dos avanços recentes em engenharia e otimização de *prompts*, o comportamento dos LLMs continua fortemente condicionado ao contexto de inferência e às ca-

racterísticas dos dados utilizados durante o pré-treinamento. Como resultado, melhorias observadas em determinados cenários nem sempre se traduzem em robustez, previsibilidade ou eficiência operacional, impondo desafios relacionados à estabilidade das respostas, capacidade de generalização e custo computacional.

- **Sensibilidade à Formulação:** Pequenas alterações na redação de instruções, na ordem de exemplos ou na organização do contexto podem modificar significativamente as distribuições de probabilidade utilizadas durante a geração. Essa elevada sensibilidade reduz a estabilidade dos *prompts* e dificulta a reprodução consistente dos resultados entre diferentes execuções, modelos ou versões de um mesmo sistema.

- **Baixa Generalização:** Desempenhos obtidos durante a otimização frequentemente dependem do modelo, domínio e conjunto de tarefas utilizados na avaliação. Em muitos casos, o *prompt* passa a explorar padrões específicos dos exemplos disponíveis, produzindo ganhos locais que não se mantêm quando aplicado a novos cenários, fenômeno análogo ao *overfitting* em aprendizado de máquina.

- **Dependência de Heurísticas:** A seleção de exemplos, o posicionamento de instruções, a definição de papéis e a estruturação do contexto são frequentemente guiados por regras empíricas e tentativa e erro. Embora essas estratégias possam melhorar o desempenho, normalmente não existem garantias formais que expliquem sua eficácia ou assegurem comportamento consistente em diferentes aplicações [Edemacu e Wu, 2025].

- **Dependência da Janela de Contexto:** Estratégias baseadas em exemplos, recuperação de documentos e raciocínio multi-etapas competem pelos mesmos recursos contextuais disponíveis durante a inferência. Mesmo em modelos com janelas extensas, a utilização eficiente desse espaço permanece um desafio, uma vez que informações relevantes podem perder influência dependendo de sua posição na sequência, fenômeno frequentemente associado ao efeito *lost-in-the-middle*.

- **Overthinking:** Cadeias de raciocínio excessivamente longas podem levar o modelo a executar etapas intermediárias redundantes mesmo quando a solução do problema já está disponível. Além de aumentar a latência e o consumo de *tokens*, esse comportamento favorece a propagação de erros ao longo da inferência e pode impedir a geração da resposta final dentro do orçamento de contexto disponível [Sui et al., 2025]. Em aplicações reais, o fenômeno impacta diretamente os custos computacionais e a eficiência operacional dos sistemas baseados em LLMs.

4.6. Considerações Finais

O capítulo apresentou técnicas de otimização de *prompts* em modelos de linguagem em larga escala, enfatizando como a formulação de instruções, exemplos e contexto influenciam diretamente o desempenho, o custo e a confiabilidade de LLMs em aplicações práticas. A análise de diferentes formas de *prompting*, de instruções e exemplos a abordagens paramétricas e de raciocínio estruturado, mostrou que ganhos em qualidade e

especialização quase sempre envolvem decisões de compromisso entre interpretabilidade e eficiência, entre flexibilidade e previsibilidade, entre maior poder expressivo e maior superfície de ataque. A avaliação de *prompts* ainda ocorre principalmente de forma indireta, via comportamento do modelo em tarefas específicas, o que dificulta mensurar sua robustez e capacidade de generalização. Ao mesmo tempo, ameaças como *prompt injection*, *jailbreaking*, *prompt leakage* e *overthinking* evidenciam que estratégias de otimização podem amplificar vulnerabilidades se não forem pensadas em conjunto com requisitos de segurança e custos operacionais.

Perspectivas promissoras de pesquisa incluem o desenvolvimento de métodos de otimização automática de *prompts* que incorporem explicitamente métricas de robustez e calibração, reduzindo a suscetibilidade a ataques adversariais sem sacrificar desempenho em tarefas-alvo. Outra direção relevante é a integração entre otimização de *prompts*, modelagem de ameaças (*threat modeling*) e arquiteturas de agentes, de modo a projetar *pipelines* de inferência que considerem, desde o início, requisitos de segurança, privacidade e auditabilidade. Há também a tendência de desenvolvimento de *benchmarks* e métricas específicos voltados à avaliação de *prompts*, e não apenas de modelos, em cenários com restrições de contexto, custo e conformidade regulatória, pavimentando o caminho para práticas de otimização de *prompts* mais sistemáticas, explicáveis e alinhadas ao uso responsável de LLMs em larga escala.

Referências

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

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A. et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[Campos et al., 2026] Campos, G. A., Andreoni, M. e Mattos, D. M. F. (2026). Avaliação da herança de propriedades adversariais em grandes modelos de linguagem com restrição de contexto e destilação de conhecimento latente. Em *Anais do SEMISH 2026 - LIII Seminário Integrado de Software e Hardware*, Gramado, RS. Sociedade Brasileira de Computação. A ser publicado.

[Chang et al., 2024] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q. e Xie, X. (2024). A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.*, 15(3).

[Chaudhari et al., 2025] Chaudhari, S., Aggarwal, P., Murahari, V., Rajpurohit, T., Kalyan, A., Narasimhan, K., Deshpande, A. e Castro da Silva, B. (2025). Rlhf deciphered: A critical analysis of reinforcement learning from human feedback for llms. *ACM Computing Surveys*, 58(2):1–37.

[Cui et al., 2025] Cui, W., Zhang, J., Li, Z., Sun, H., Lopez, D., Das, K., Malin, B. A. e Kumar, S. (2025). Automatic prompt optimization via heuristic search: A survey. *arXiv preprint arXiv:2502.18746*.

[Du et al., 2026] Du, S., Zhao, J., Shi, J., Xie, Z., Jiang, X., Bai, Y. e He, L. (2026). A survey on the optimization of large language model-based agents. *ACM Computing Surveys*, 58(9):1–37.

[Edemacu e Wu, 2025] Edemacu, K. e Wu, X. (2025). Privacy preserving prompt engineering: A survey. *ACM Computing Surveys*, 57(10):1–36.

[Jain e Jindal, 2025] Jain, A. M. e Jindal, M. (2025). Systematic survey of various prompt optimization methods and their classifications. Em *2025 11th International Conference on Computing and Artificial Intelligence (ICCAI)*, p. 524–536. IEEE.

[Li, 2023] Li, Y. (2023). A practical survey on zero-shot prompt design for in-context learning. Em *Proceedings of the 14th international conference on recent advances in natural language processing*, p. 641–647.

[Li et al., 2025a] Li, Z., Liu, Y., Su, Y. e Collier, N. (2025a). Prompt compression for large language models: A survey. Em Chiruzzo, L., Ritter, A. e Wang, L., editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, p. 7182–7195, Albuquerque, New Mexico. Association for Computational Linguistics.

[Li et al., 2025b] Li, Z., Su, Y. e Collier, N. (2025b). A survey on prompt tuning. *arXiv preprint arXiv:2507.06085*.

[Liu, 2025] Liu, Z. (2025). Reinforcement learning for prompt optimization in language models: A comprehensive survey of methods, representations, and evaluation challenges. *ICCK Transactions on Emerging Topics in Artificial Intelligence*, 2(4):173–181.

[Moia et al., 2026] Moia, V. H. G., Sanz, I. J., Rebello, G. A. F., de Meneses, R. D., Hitaj, B. e Lindqvist, U. (2026). Llm in the middle: A systematic review of threats and mitigations to real-world llm-based systems. *Computer Science Review*, 61:100916.

[Saleem et al., 2026] Saleem, S., Asim, M. N., Zulfiqar, S. e Dengel, A. (2026). The evolution of natural language processing: How prompt optimization and language models are shaping the future. *Computer Science Review*, 61:100938.

[Sandoval et al., 2023] Sandoval, G., Pearce, H., Nys, T., Karri, R., Garg, S. e Dolan-Gavitt, B. (2023). Lost at c: A user study on the security implications of large language model code assistants. Em *32nd USENIX Security Symposium (USENIX Security 23)*, p. 2205–2222, Anaheim, CA. USENIX Association.

[Sharma et al., 2025] Sharma, R., Mehta, M. e Raina, S. T. (2025). Rlhf: A comprehensive survey for cultural, multimodal and low latency alignment methods. Em *International Conference on Computing and Communication Networks*, p. 230–246. Springer.

[Singla et al., 2025] Singla, A., Sukharevsky, A., Hall, B., Yee, L., Chui, M. e Balakrishnan, T. (2025). The state of AI in 2025: Agents, innovation, and transformation. Technical report, QuantumBlack, AI by McKinsey.

[Sui et al., 2025] Sui, Y., Chuang, Y.-N., Wang, G., Zhang, J., Zhang, T., Yuan, J., Liu, H., Wen, A., Zhong, S., Zou, N. et al. (2025). Stop overthinking: A survey on efficient reasoning for large language models. *arXiv preprint arXiv:2503.16419*.

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

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

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

[Zhang et al., 2025] Zhang, C., Zhou, L., Xu, X., Wu, J. e Liu, Z. (2025). Adversarial attacks of vision tasks in the past 10 years: A survey. *ACM Computing Surveys*, 58(2):1–42.

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

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

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