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].

 DOI: 10.5753/sbc.20175.9 ©2026 SBC

- ***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,

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.