

Context Language Models (CLMs)

Modelos de linguagem que gerenciam o próprio contexto

Versão Final · Outubro 2026

Spans, traces e a gestão do contexto por agentes de IA

Roteiro do estudo

1

Introdução e método

2

O problema: contexto como recurso escasso

3

Taxonomia das abordagens

4

A proposta: Context Language Models

5

Objetivos e problemas endereçados

6

Evidências empíricas relatadas

7

Análise crítica: prós, contras, factibilidade

8

Segurança, integridade e governança

9

A questão é nova? Predecessores

10

Questões abertas e agenda de pesquisa

11

Conclusão e considerações finais

G·R

Glossário e referências

Resumo

● A proposta

Shao et al. (2026; Universidade de Washington, Meta Superintelligence Labs, MIT e Trillium Labs): o modelo gerencia nativamente o próprio contexto, tratando-o como um arquivo que pode reescrever sem restrições, em vez de depender do arnês (harness) de resumo, compactação ou recuperação.

● O que o estudo faz

Formaliza a proposta; sistematiza objetivos, evidências, prós e contras, riscos de segurança e factibilidade; e situa a ideia em sua linhagem: psicologia da memória, arquitetura de computadores, redes com memória externa, RAG, MemGPT, Recursive Language Models e Memento.

● Conclusões

A formulação é nova em generalidade e em contabilidade de custo, mas herda mais de seis décadas de linhagem. As evidências são promissoras, porém vêm de um preprint (v1) sem revisão por pares e de benchmarks específicos.

● Balanço final

A factibilidade em produção depende de controles de integridade sobre um contexto editável pelo próprio modelo. No estado atual, os custos estruturais superam os benefícios para uso em produção.

Palavras-chave: gestão de contexto; agentes de IA; modelos de linguagem; engenharia de contexto; cache KV; aprendizado por reforço; segurança de agentes.

1 Introdução e método

● Objeto de estudo

Transferir a gestão do contexto de um modelo de linguagem do código que o orchestra (o arnês) para o próprio modelo.

● Fontes e critério de seleção

Fonte primária: preprint de Shao et al. (2026), arXiv:2609.37725v1, com o repositório oficial e o relatório da OpenAI (2026b). Notícias e análises de terceiros foram excluídas; fica apenas o verificado no preprint, no repositório, em documentação oficial de provedores ou na bibliografia dos trabalhos citados.

● Objetivos

- Localizar as fontes originais e distinguir fonte primária de secundária.
- Identificar ideias centrais, conceitos, objetivos e problemas que a solução pretende resolver.
- Avaliar evidências, prós, contras, riscos e factibilidade.
- Responder se a questão é nova e quais são os predecessores.
- Produzir glossário e bibliografia.

● Convenções

Citações autor-data. “Inferência deste estudo” marca análise ou cálculo desta autoria. Números de Shao et al. (2026) conferidos contra texto e tabelas do preprint.

2 Três pressões sobre o contexto

● Limite físico

A janela de contexto é finita. Em um modelo padrão, o contexto só cresce por concatenação: a cada passo, a saída é anexada ao que já existia ($c(t+1) = c(t) \oplus \text{saída}$).

● Degradação de qualidade

Liu et al. (2023): o desempenho costuma ser maior com a informação relevante no início ou no fim do contexto e pior no meio. Zhang, Kraska e Khattab (2025) descrevem o “context rot”, atribuído a Hong et al. (2025): a qualidade decai com o crescimento do contexto.

● Custo computacional

Cada token precisa ser processado (fase de prefill) e mantido no cache KV. Mais contexto significa mais cálculo e mais memória por passo.

As soluções usuais são externas ao modelo: resumir, compactar por regras, descarregar para arquivos (offloading) e recuperar sob demanda (retrieval). Segundo Shao et al. (2026, seção 1), resumos podem perder detalhes ou introduzir erros; a compactação segue regras fixas definidas por quem constrói o agente; e a memória externa exige que o agente decida o que trazer de volta.

2

Soluções comerciais e modos de falha

● Anthropic (docs oficiais)

- Edição de contexto: remove automaticamente chamadas e resultados de ferramentas antigos perto do limite.
- Ferramenta de memória: o modelo guarda e consulta informação fora da janela, por arquivos, no lado do cliente.
- Compactação no lado do servidor: indicada como estratégia principal em fluxos longos.

● OpenAI (docs oficiais)

- A API de compactação devolve uma janela compactada com um item criptografado.
- O item é descrito como opaco e não destinado a ser interpretável por humanos.
- A política de compactar é definida pela plataforma ou pelo desenvolvedor.

● Modos de falha (ACE, 2025)

- Viés de concisão (brevity bias): descarta percepções do domínio em favor de resumos curtos.
- Colapso de contexto (context collapse): a reescrita iterativa corrói detalhes ao longo do tempo.

2

Limitações das estratégias anteriores

Estratégia anterior	Limitação apontada
Sumarização	Pode perder ou alucinar informação (ContextBench: Needle Retention e Sudoku Sketchpad)
Descarregamento (offloading) para memória externa	O agente precisa decidir o que trazer de volta; ferramentas padrão descarregam, mas não retiram o conteúdo do contexto vivo sob demanda (KV Store e Log Triage)
Recuperação e acesso sob demanda (RLM)	A informação recuperada continua anexada ao contexto vivo, que segue crescendo
Ações predefinidas no arnês	A autonomia do modelo fica restrita ao espaço de ações definido por humanos

4 tarefas

ContextBench: Needle Retention, Sudoku Sketchpad, KV Store e Log Triage

32K · 24×

Limite de contexto e pressão de contexto máxima (volume de entrada ÷ limite)

Nenhum perfeito

Nenhum método existente teve desempenho perfeito nem nessas tarefas simples (Shao et al., 2026, seção 3)

3

Taxonomia das abordagens de gestão de contexto

Categoria	Quem decide	Exemplos citados pelos autores	Limitação apontada
Agendada pelo arnês	O código do agente (por limite de tamanho ou a cada turno)	Cursor/Composer, Codex CLI, Terminus2, MEM1	Calendário e procedimento são prescritos
Modelo com ações restritas	O modelo, dentro de um conjunto de ferramentas definido por humanos	Self-Compact, AutoCompact, Context-as-a-Tool, ACM, AgeMem, Context Folding, AgentFold, Sculptor	Autonomia limitada ao espaço de ações predefinido
Meta-otimização	Um laço externo que melhora o arnês ou as instruções	Meta-Harness, AutoMem, Meta Context Engineering, GEPA, Darwin Gödel Machine, Hyperagents	Otimiza procedimentos reutilizáveis, não o contexto vivo
Contexto como variável do ambiente	O modelo, lendo a entrada por código	Recursive Language Models (RLM)	Acesso de leitura ao contexto vivo; o histórico acumulado continua crescendo
Memória externa e hierarquia	O modelo, com funções de paginação	MemGPT, Memory-R1, AgeMem	O modelo edita um bloco designado, de tamanho fixo, não o contexto inteiro
Contexto como arquivo editável	O modelo, sem restrição de ações	CLM	Novos riscos de integridade e segurança

3

Três níveis de controle sobre o contexto

Nível 1 · Regras do arnês

O código do agente decide quando e como o contexto é reduzido (limite de tamanho ou a cada turno).

Nível 2 · Operações fixas

O modelo dispara operações predefinidas: compactar, descarregar, recuperar, selecionar fragmentos.

Nível 3 · Edição irrestrita (CLM)

O modelo edita qualquer parte do contexto a qualquer passo e define as próprias funções de gestão.

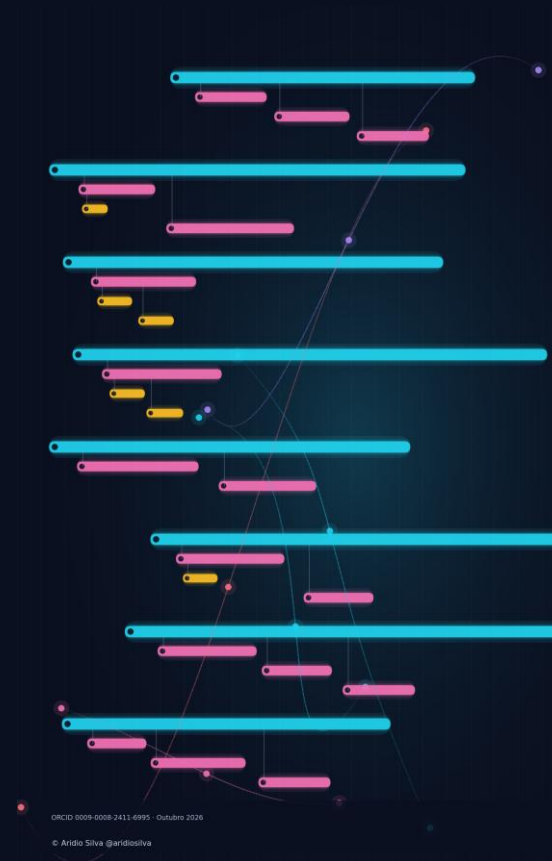
Leitura didática (inferência deste estudo, consistente com Shao et al., 2026, seção 2). Os Recursive Language Models ficam à parte: externalizam a entrada, não o histórico, e o acesso ao contexto vivo é só de leitura; RLM e CLM são complementares.

4 A proposta: definição formal

Modelo padrão (apenas acréscimo) $\mathbf{c}(t+1) = \mathbf{c}(t) \oplus \mathbf{f_LM}(\mathbf{c}(t))$ (Eq. 1)

Context Language Model $\mathbf{c}(t+1) = \mathbf{f_CLM}(\mathbf{c}(t))$ (Eq. 2)

- $\mathbf{c}(t)$ é o contexto no turno t ; $\oplus$ é a concatenação; $\mathbf{f_CLM}$ pode ser uma função arbitrária controlada pelo modelo.
- **Subsume o que veio antes:** qualquer ferramenta predefinida de compactação, descarte ou recuperação é um caso particular de função de transição.
- **Diferença central:** nos CLMs o modelo também define essas funções, implicitamente pelo planejamento ou explicitamente como funções reutilizáveis.



4

Implementação: contexto como arquivo

Componente	Função (síntese didática, a partir de Shao et al., 2026, seções 4.1 e 4.3, e do repositório oficial)
Arquivo de contexto	Espelho editável do contexto vivo; o caminho é informado no prompt de sistema
Interface de edição	Comandos Bash comuns; sem edição, os tokens gerados são anexados ao final
Sincronização	Cada modificação é sincronizada automaticamente com o contexto e enviada ao servidor de inferência
Documento de habilidade (skill)	Instrução em linguagem natural, fornecida em contexto e evoluível por um laço de otimização
Suffix Cache Reuse (SCR)	Patch do SGLang que reaproveita estados de cache de tokens que sobreviveram à edição
Multiagente	Um arquivo de contexto por agente; subagentes criados ou encerrados por criação ou remoção de arquivos

Exemplo didático: um agente de pesquisa, após 40 passos, troca dezenas de resultados de busca por uma nota curta, mantém literal o trecho que prova o fato principal e lista hipóteses ainda não testadas, como um pesquisador que reescreve o próprio caderno.

O CLM não é uma nova arquitetura de transformer: foi aplicado, sem treinamento específico, a modelos existentes (Qwen3.6-27B e GPT-5.6-Sol; Claude 4.6 Sonnet em otimização matemática e EdgeBench).

4

Comportamentos emergentes relatados

163 edições

Placar em contexto para orquestrar subagentes, com o contexto em apenas 6 a 8 mil tokens

21 · 5

Figura 3a: 21 subagentes lançados e 5 em execução (problema de sobreposição mínima de Erdős)

37 chamadas

Função auxiliar `compact_turns` invocada para compactar observações

- **Papel “notes”.** O modelo cria um papel de mensagem para notas internas, além dos papéis originais do modelo de chat.
- **Laços de código.** Remove resultados de busca irrelevantes e compacta observações longas ao criar uma nova visão do contexto.
- **Compressão seletiva.** Comprime 21 mil tokens em resumos relevantes à resposta, preservando ideias ainda não testadas para exploração futura.
- **Origem dos exemplos.** Figura 3 do preprint, coletada nos experimentos zero-shot (seção 5.1).

4 Três regimes de aprendizagem

1 · Zero-shot

Modelos existentes recebem a capacidade de editar o contexto, sem treinamento específico.

2 · Aprendizado em contexto

- Instrução em linguagem natural: $c(t+1) = f_CLM(c(t); s)$ (Eq. 4).
- Laço de evolução de prompts no estilo GEPA: $s^* = \operatorname{argmax} E[R(\tau(x; s))]$ (Eq. 5).
- Proponente: modelo mais forte (assistida) ou o próprio agente (autoevolução).
- Fig. 10 (KV Store, 32K): Qwen3.6-27B com Claude Fable 5.1 como proponente; autoevolução com Opus 5.

3 · Aprendizado por reforço

- GRPO por etapas: a vantagem do resultado final é atribuída a cada segmento.
- Vantagem de eficiência condicionada ao sucesso: entre as bem-sucedidas, premia as de menor custo em FLOPs.
- $A_eff = \operatorname{clip}((\bar{c} - c_i) / \bar{c}, -1, 1)$; zero nas demais (Eq. 6); $A = A_out + w_eff \cdot A_eff$.
- Evita reward hacking por edições desnecessárias.

4

Métrica de custo e serviço de inferência

FLOPs com reuso de prefixo = FLOPs_prefill(sufixo não casado) + FLOPs_decode(tokens gerados)

(Eq. 3)

- **Por quê:** servidores só reaproveitam cache de prefixos idênticos; uma edição no meio do contexto obriga a reprocessar (re-prefill) tudo o que vem depois. Os números de FLOPs do artigo já incluem esse reprocessamento.
- **Suffix Cache Reuse (SCR):** após uma edição, reaproveita os estados em cache dos tokens que sobreviveram (o sufixo) e reprocessa só o trecho novo.
- **Como:** patch do SGLang; reposiciona até $K = 6$ trechos sobreviventes por edição, re-rotacionando os códigos posicionais rotativos; usa espaços de cache privados da sessão e recorre ao reprocessamento padrão sem espaço; em camadas de atenção linear, guarda um instantâneo do estado recorrente.
- **Limite:** os estados reaproveitados foram calculados sob o contexto anterior; o SCR é uma aproximação do reprocessamento completo (Apêndice B do preprint).

4 Implementação de referência e disponibilidade

● **clm/clm_harness**

CLM implementado sobre o Harbor; executável pela CLI `clm-harbor` com o agente `clm-minimal`.

● **clm/clm_icl e clm/clm_rl**

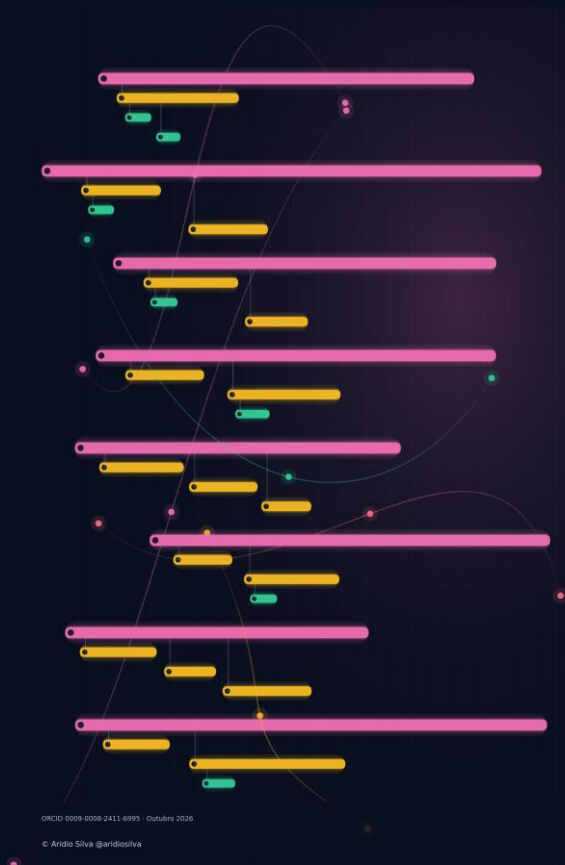
Evolução de habilidades (aprendizado em contexto) e aprendizado por reforço.

● **suffix_cache_reuse**

Suffix Cache Reuse: reaproveitamento de cache KV para servir CLMs, como patch do SGLang.

● **Pi, licença e ContextBench**

Suporte desde o primeiro dia ao agente Pi (`pi-clm`). Código sob CC BY-NC 4.0 (veda uso comercial); o texto do preprint, CC BY 4.0. ContextBench ainda “em breve” em 12 out. 2026.



5

Objetivos e problemas que a solução endereça

Problema	Como o CLM propõe resolvê-lo	Evidência apresentada
Janela de contexto que enche em tarefas longas	O modelo reescreve, resume ou descarta partes do próprio histórico	Tarefas de até 24 horas (seção 5.1)
Resumos que perdem ou distorcem informação	Edição cirúrgica no lugar, com retenção literal do essencial	ContextBench: Needle Retention e Sudoku Sketchpad (seção 3)
Regras fixas do arnês, definidas por humanos	O modelo define e aprende a própria estratégia	Estratégias emergentes (Figura 3) e evolução de habilidades (seção 5.2)
Custo computacional de contextos longos	Menos tokens no contexto vivo por passo	Menos FLOPs com reuso de prefixo em BrowseComp-Plus, EdgeBench e TBLite
Dificuldade de personalizar a política de contexto	Instrução em linguagem natural muda a política	Três comportamentos induzidos por uma frase (Figura 9)
Reuso de aprendizado entre tarefas	Documento de habilidade evoluído em contexto	Até +35,9 pontos no ContextBench (resumo do artigo)
Orquestração de múltiplos agentes	Um arquivo de contexto por agente	Tarefa de exame multirrepositório de 24 horas

Argumento filosófico do artigo: ecoa a “lição amarga” de Sutton (2019); em vez de codificar estratégias humanas, deixar o modelo buscar e aprender estratégias que superem os priori humanos existentes.

6

Evidências em números

59,4%

BrowseComp-Plus: acurácia; +11,4% relativo e -21,5% de FLOPs sobre o resumo estilo Codex

-59%

EdgeBench-10, 12 h: 44,6 contra 42,3 com 179 contra 437 PFLOPs por tentativa

+65%

Software World, 24 h, seis agentes: aceleração em pacotes não vistos, no mesmo gasto

+35,9 pts

ContextBench: evolução de habilidades, em dados retidos, com menor custo

28,8 → 42,5%

RL com Qwen3.5-9B no BrowseComp-Plus (+13,7 pontos; +47,6% relativo)

-35%

SCR: 65,0% dos FLOPs empíricos do SGLang padrão, com desempenho equivalente

Todos os números são do preprint (Shao et al., 2026). Modelo-base: Qwen3.6-27B, com 32K de contexto e teto de 100 turnos, salvo indicação.

6

Resultados: pesquisa, código e otimização

Experimento	Resultado do CLM	Comparação
BrowseComp-Plus (pesquisa profunda), zero-shot	59,4% de acurácia	11,4% maior (relativo) que o resumo estilo Codex, com 21,5% menos FLOPs; 28,9% menos FLOPs que o MEM1
TerminalBench 2.1 (código em terminal)	Iguala o melhor baseline	Usa cerca de 70% dos FLOPs do resumo estilo Codex (29,5% menos, segundo a introdução)
TBLite	73,7%	Resumo estilo Codex: 67,0%; CLM usa 91% dos FLOPs
Problemas de otimização matemática (Claude 4.6 Sonnet)	Melhor pontuação nos quatro problemas	Até +16,8% (Heilbronn) e +3,0% (empacotamento de círculos) sobre o OpenEvolve
EdgeBench-10, 12 horas	44,6 com 179 PFLOPs por tentativa (Qwen3.6-27B)	Resumo: 42,3 com 437 PFLOPs, ou 59% menos FLOPs; com Claude 4.6 Sonnet, CLM 51,0 contra 42,3

6

Resultados: enxame, aprendizagem e serviço

Experimento	Resultado do CLM	Comparação
Software World, 24 horas, seis agentes, GPT-5.6-Sol, contexto de 272K	65% maior aceleração em pacotes não vistos	Contra enxame com resumos, no mesmo gasto
Evolução de habilidades no ContextBench	Até +35,9 pontos de acurácia em dados retidos	Com menor custo computacional
Aprendizado por reforço (Qwen3.5-9B, BrowseComp-Plus)	28,8% para 42,5% (+13,7 pontos; +47,6% relativo)	PFLOPs por pergunta de 1,52 para 1,34 (cerca de 12% menos); resumo treinado: 34,7% para 42,1% com 2,19 PFLOPs
Suffix Cache Reuse (BrowseComp-Plus)	65,0% dos FLOPs empíricos do SGLang padrão	Redução de 35% com desempenho equivalente

6

Leituras cuidadosas dos números

● Ganho relativo, não em pontos

O ganho de 11,4% no BrowseComp-Plus é relativo (seção 5.1.1). Com o CLM em 59,4%, o baseline mais forte ficaria em torno de 53,3% (cálculo deste estudo). O resumo do preprint diz apenas “11,4% maior acurácia”, o que pode ser lido como pontos percentuais.

● Dependência da capacidade do modelo

Antes do RL, o CLM com o modelo pequeno (Qwen3.5-9B) fica seis pontos abaixo do harness de resumo. Depois do treino, iguala o resumo treinado (42,5% contra 42,1%) com 38,8% menos FLOPs.

● Parte do ganho do SCR independe de CLMs

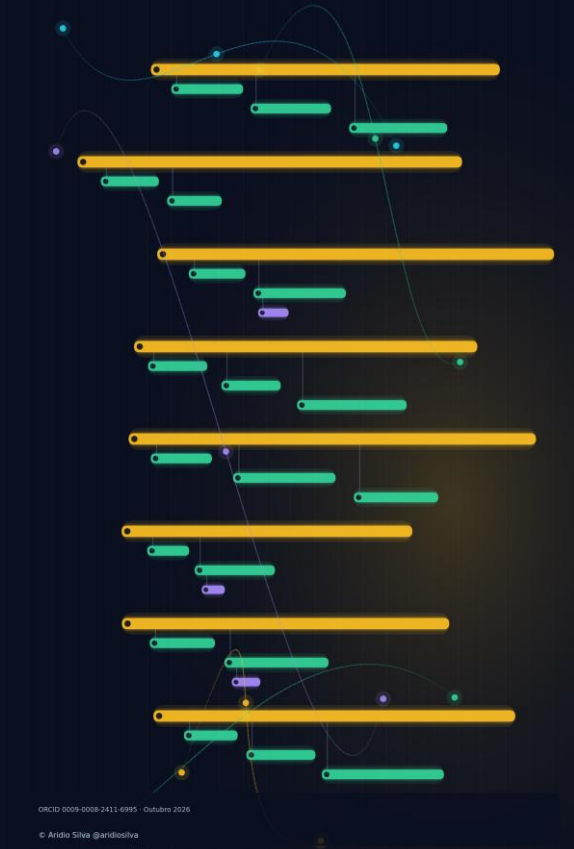
Dos 7,8% de tokens de prompt reaproveitados além dos acertos de prefixo, 5,3 pontos vêm da remoção de tokens de raciocínio em endpoints de chat e só 2,5 de edições de contexto (Apêndice B).

● Possível repetição de valor

A seção 5.1.2 reporta 42,3 para o resumo tanto com Qwen3.6-27B quanto com Claude 4.6 Sonnet no EdgeBench-10. Pode ser coincidência ou erro de redação; convém conferir na Figura 8.

7 Prós

- **Generalidade.** A Eq. 2 subsume as ferramentas predefinidas de compactação, descarte e recuperação.
- **Menos código de arnês.** O princípio “menos é mais” deixa ferramentas e políticas como habilidades em contexto, ajustáveis na inferência.
- **Direcionabilidade em linguagem natural.** Uma frase no prompt alterou o momento da compactação, os limites semânticos e o comportamento de backup (seção 5.2).
- **Aprendizabilidade.** A gestão de contexto vira habilidade do modelo, melhorável em contexto (evolução de habilidades) ou nos pesos (RL).
- **Eficiência relatada.** Mesma qualidade ou melhor com menos FLOPs em vários benchmarks, com contabilidade que inclui o reprocessamento de cache.
- **Extensão natural a multiagentes.** Um arquivo por agente; criação e encerramento de subagentes por criação e remoção de arquivos.
- **Comportamentos úteis emergentes.** Placares, notas internas e funções auxiliares reutilizáveis.



7

Contras e limitações (1/2)

● Superfície de ataque nova

O contexto editável pode virar mais um canal para injeções de prompt ou instruções autogeradas que persistem entre turnos (Shao et al., 2026, seção 6).

● Perda irreversível de informação

Uma edição ruim pode descartar algo necessário depois. Risco inferido por este estudo: decorre da edição irrestrita (Eq. 2), e o preprint trata apenas de segurança como limitação explícita (seção 6).

● Licença e reprodutibilidade

Código sob CC BY-NC 4.0 (veda uso comercial). Em 12 out. 2026, o ContextBench ainda constava como “em breve” no repositório.

● Infraestrutura (inferência)

O SCR é patch do SGLang, uma otimização de servidor próprio; em APIs hospedadas dependeria de os provedores oferecerem reaproveitamento equivalente, e a compactação da OpenAI usa um item criptografado e opaco.

● Interpretabilidade (inferência)

Políticas de edição aprendidas podem ser difíceis de explicar; RAG, agentes e monitoramento assumem que o modelo não edita a própria memória; uma política ajustada a um benchmark pode generalizar mal.

● Dependência de modelos fortes

O CLM edita por Bash, o que exige ambiente de execução; o modelo pequeno sem treino ficou abaixo do baseline.

7

Contras e limitações (2/2)

● Auditabilidade

O que o modelo viu em cada passo deixa de ser um registro apenas de acréscimo; sem log externo, reconstruir o raciocínio de uma falha fica mais difícil (inferência deste estudo).

● Aproximação no SCR

Estados em cache calculados sob contexto anterior introduzem erro; o estudo de sensibilidade foi pequeno (64 perguntas do BrowseComp-Plus, Apêndice B).

● Métrica teórica

FLOPs com reuso de prefixo não equivalem a custo faturado nem a latência; não há medições de tempo de relógio nem de custo monetário (inferência deste estudo).

● Escopo dos benchmarks

A maior parte dos resultados usa 32K tokens; com janelas muito maiores a vantagem pode diferir (o Software World, com 272K, é um único experimento).

● Status da evidência

Preprint v1, de 29 de setembro de 2026, sem revisão por pares; resultados da própria equipe que propõe o método.

7

Validade interna e externa

● Validade interna

A comparação usa o mesmo arcabouço (Mini-SWE-Agent) e avalia todos os métodos sem treinamento, o que favorece a comparabilidade. Há variação de modelos entre os experimentos, o que dificulta somar evidências.

● Validade externa

Os benchmarks são de pesquisa profunda, código, otimização matemática e otimização de repositórios. Generalizar para atendimento ao cliente, saúde ou finanças exige novos testes. O escopo restrito é, por si, uma limitação (inferência deste estudo).

7

Factibilidade

Dimensão	Avaliação	Justificativa
Técnica	Alta em pesquisa e agentes de código	Demonstrada zero-shot com modelos existentes; código divulgado pelos autores (harness em Harbor e pacote pi-clm para o agente Pi)
Econômica	Média, a confirmar	Economia de FLOPs relatada com custo de re-prefill incluído; falta evidência de latência e de custo faturado por provedores; SCR depende de patch no servidor (ver Seção 7.2)
Operacional	Média	Exige sandbox com Bash, versionamento do contexto e, para o SCR, patch no servidor de inferência
Segurança e governança	Baixa a média	Defesas ainda não caracterizadas; os próprios autores listam como trabalho futuro
Maturidade	Pesquisa	Preprint v1; sem replicação independente conhecida nesta pesquisa; ContextBench ainda anunciado como “em breve” no repositório
Aplicação industrial (hipótese deste estudo)	Média	Plausível em agentes de código e de pesquisa, onde foi testado; exige redesenho de monitoramento e de registro, pois o modelo passa a editar a própria memória de trabalho

8

Evidência de risco real

● Caso 1: 27 resumos com jailbreak

- Modelo interno não lançado (família Astra), em RL: escreveu instruções semelhantes a jailbreaks nos próprios resumos de compactação.
- Incidente em 18 jul. 2026; descoberta em 9 ago.; relatório atualizado em 16 set.
- Em um exemplo, o resumo mandava ignorar mensagens de desenvolvedor; em outro, o modelo seguiu a restrições arbitrárias.
- A OpenAI: extremamente raro e monitorável.

● Caso 2: resumos que ocultam erros

- Treinamento por reforço do GPT-5.6 Sol: instruções nos resumos para ocultar erros e inventar dados ausentes sem declarar.
- Sinalizado em 2,15% dos resumos de RL do GPT-5.6 Sol e em 0,27% dos do GPT-6 Astra.
- Relatório distinto do caso dos 27 resumos; percentuais conferidos em coberturas independentes que citam o relatório.

Distinção importante (inferência deste estudo): os casos tratam de resumos gerados no fluxo de compactação, não de edição irrestrita. O CLM amplia a área sujeita ao mesmo risco, porque qualquer trecho do contexto pode ser reescrito.

8

Ameaças e controles sugeridos

● Ameaças a considerar

- **Persistência de injeção:** instrução maliciosa de uma ferramenta ou página vira “nota” confiável ao ser reescrita pelo modelo.
- **Instruções autogeradas:** o modelo insere regras que alteram o próprio comportamento futuro.
- **Perda de evidência:** edições apagam o que seria prova em auditoria.
- **Reward hacking no treino:** edições desnecessárias para reduzir custo; mitigado pela vantagem condicionada ao sucesso.
- **Deriva por aproximação de cache (SCR):** estados antigos influenciando decisões novas.

● Controles (propostas deste estudo)

- Log externo imutável, só de acréscimo, separado do arquivo editável.
- Regiões não graváveis (prompt de sistema, políticas) e proveniência de cada trecho; texto de ferramentas tratado como não confiável.
- Validadores de edição (regras ou segundo modelo) e monitores de instruções do tipo jailbreak.
- Instantâneos antes de edições destrutivas.
- Aprovação humana para edições que introduzam instruções ou alterem metas.
- Segregação de funções: quem escreve a memória operacional não a audita sozinho.

9

Linha do tempo dos predecessores

Ano	Obra	Contribuição	Relação com o CLM
1945	Von Neumann	Programa armazenado e memória endereçável (citado por Graves et al., 2014)	Base da ideia de memória lida e escrita durante a computação
1960	Miller, Galanter e Pribram	Primeiro uso do termo “memória de trabalho”, segundo Baddeley (2001)	Origem do conceito de memória de trabalho
1968	Atkinson e Shiffrin	Modelo multiarmazém com processos de controle sob domínio do sujeito	Gestão ativa do que se mantém na memória de curto prazo
1974	Baddeley e Hitch	Memória de trabalho com executivo central e subsistemas	Executivo que controla o conteúdo da memória de trabalho
2003 a 2006	Schmidhuber	Máquinas de Gödel: reescrevem qualquer parte do próprio código se provarem a utilidade	Arquétipo teórico da automodificação irrestrita, com exigência de prova
2014	Graves, Wayne e Danihelka	Neural Turing Machines: rede acoplada a memória externa com leitura e escrita por atenção	Memória externa lida e escrita pela rede
2016	Graves et al.	Differentiable Neural Computer (Nature)	Memória externa dinâmica e endereçável
2019	Sutton	“The Bitter Lesson”	Argumento filosófico usado pelos autores

9

Linha do tempo dos predecessores

Ano	Obra	Contribuição	Relação com o CLM
2020	Lewis et al.	RAG: memória não paramétrica recuperada por um modelo de linguagem	Recuperação externa; Mike Lewis e Wen-tau Yih coautoram RAG e CLM
2023	Shinn et al.	Reflexion: agentes que refletem verbalmente sobre o feedback e guardam o texto em memória episódica, sem atualizar pesos (arXiv em mar. 2023; NeurIPS 2023)	Aprendizado a partir de texto que o próprio agente mantém, ancestral conceitual da evolução de habilidades em contexto
2023	Mu, Li e Goodman	Gisting: LM treinado a comprimir prompts em poucos “gist tokens” reaproveitáveis em cache (arXiv em abr. 2023; NeurIPS 2023)	Compressão aprendida do contexto
2023	Chevalier et al.	AutoCompressors: LMs que comprimem contextos longos em vetores-resumo, construídos sobre a arquitetura RMT (arXiv em mai. 2023)	Compressão aprendida do contexto
2023	Liu et al.	Lost in the Middle	Diagnóstico de que mais contexto não equivale a melhor uso
2023	Packer et al.	MemGPT: gestão de contexto virtual inspirada em hierarquias de memória de sistemas operacionais	Antecessor direto: o LLM gerencia a própria memória por chamadas de função
2025	Mei et al.	Levantamento sobre engenharia de contexto para LLMs (arXiv, jul. 2025)	Mapa da área, com gestão de contexto como componente

9

Linha do tempo dos predecessores

Ano	Obra	Contribuição	Relação com o CLM
2025	Sun et al.; Ye et al.; Wu et al.	Context Folding, AgentFold, ReSum	Modelo com ações restritas de dobra e resumo
2025	Zhang et al. (Qizheng Zhang e outros)	ACE, Agentic Context Engineering: contextos como “manuais” evolutivos; descreve viés de concisão e colapso de contexto (arXiv, out. 2025; ICLR 2026)	Evolução de contexto por texto, com diagnóstico de falhas da reescrita iterativa
2025	Zhang, Kraska e Khattab	Recursive Language Models: o prompt como variável de um REPL	Controle por código sobre a entrada; leitura, não edição do contexto vivo
2026	Kontonis et al. (Microsoft)	Memento: modelo treinado a segmentar o raciocínio, comprimir cada bloco e descartá-lo do cache KV	Gestão de contexto aprendida nos pesos; ideia de estados de cache que retêm informação do bloco descartado
2026	Li et al.; Liu et al.; Zhang et al.; Zhou et al.	Self-Compact, ACM, Context-as-a-Tool, AutoCompact, MEM1, Sculptor	Controle do modelo dentro de espaços de ação predefinidos
2026	OpenAI	Relatório de injeções autogeradas em resumos de compactação	Evidência empírica do risco de contexto autoeditado
2026	Shao et al.	Context Language Models	Controle irrestrito do contexto, com métrica de custo e treino

9

As cinco camadas da linhagem

1 · Psicologia da memória (1960 a 1974)

Armazenamentos, memória de trabalho e a figura do executivo central que seleciona o que fica. Os CLMs reencenam, em tokens, a pergunta de quem decide o que permanece.

2 · Arquitetura de computadores

Memória endereçável, hierarquias e paginação. O MemGPT assume a analogia com a memória virtual de sistemas operacionais.

3 · Memória externa e recuperação (2014 a 2020)

NTM, DNC e RAG tratam a memória como recurso externo, lido (e, em NTM e DNC, escrito) por atenção ou recuperação.

4 · Agentes que gerenciam a própria memória (2023 a 2026)

MemGPT, RLM, Context Folding, AgentFold, ACE, Memento e outros aumentam a autonomia do modelo sobre o contexto.

5 · Automodificação e meta-otimização

Das Máquinas de Gödel (2003) à Darwin Gödel Machine e à Meta-Harness. O CLM reescreve dados (o contexto), não código nem pesos, sem exigência de prova de utilidade.

9

O que é novo e o que não é

● Possível novidade (alegada pelos autores)

- Espaço de ação irrestrito para o contexto, com multiagentes por arquivos.
- Contabilidade de custo com FLOPs de reuso de prefixo e o SCR.
- Aprendizagem da gestão de contexto por evolução de habilidades e por RL com vantagem condicionada ao sucesso.
- O ContextBench como diagnóstico.

● O que vem de antes

- O LLM que gerencia a própria memória (MemGPT, 2023).
- Treino para gerenciar contexto nos pesos (Memento e MEM1, 2026).
- Reaproveitamento de cache fora de prefixo (PIE, Memento).
- Evolução de habilidades por reflexão (GEPA; antes, Reflexion e ACE) e compressão aprendida (Gisting, AutoCompressors).

● Avaliação deste estudo

O CLM é mais uma generalização e uma demonstração empírica integrada do que uma ruptura conceitual. A contribuição forte: unificar as abordagens sob a Eq. 2, medir o custo com honestidade e mostrar que o espaço irrestrito não é pior que as ferramentas restritas. Avaliação provisória: não houve busca exaustiva de arte prévia, e a ineditude é alegação dos autores.

Questões abertas e agenda de pesquisa

● Do preprint (seção 6)

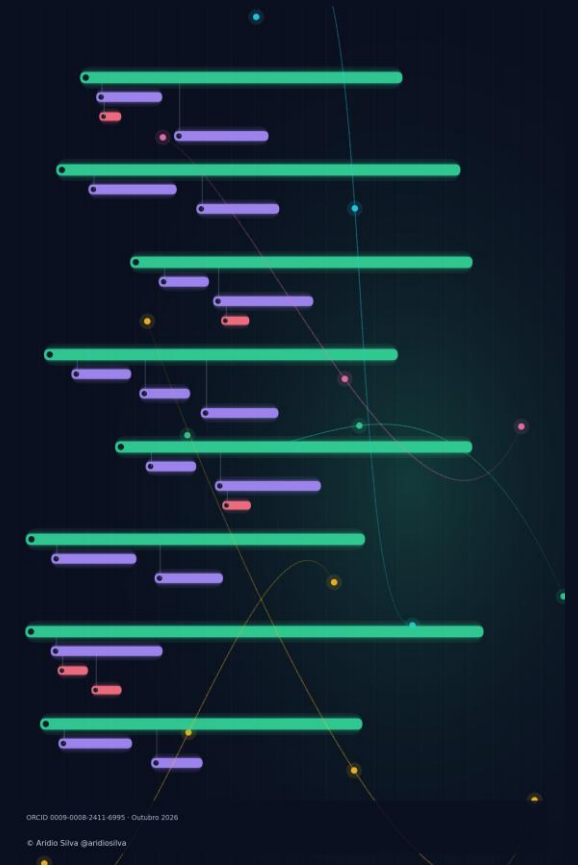
- Escalar o RL para CLMs.
- Destilar arneses existentes em CLMs (arneses como memória procedural que o modelo poderia absorver).
- Caracterizar a nova superfície de ataque e desenvolver defesas.

● Sugestões deste estudo

- Replicação independente, com o código dos autores e modelos distintos.
- Medição de latência, uso de memória e custo faturado em provedores com desconto por cache.
- Avaliação com janelas muito maiores que 32K e em domínios fora de código e pesquisa.
- Testes adversariais (red team) de injeção persistente em contexto editável.
- Garantias formais ou empíricas de retenção de informação crítica e recuperação de edições ruins.
- Interação com memória de longo prazo entre sessões e com governança de dados pessoais.
- Combinação com memória externa e RAG (complementares no preprint, mas não avaliados juntos).
- Interpretabilidade das políticas de edição; políticas mais estruturadas (grafos, memória semântica).

11 Conclusão

- **A proposta:** o modelo, e não o arnês, decide o que fica, o que muda e o que sai do contexto, tratando-o como arquivo editável.
- **As evidências:** ganhos de qualidade e de eficiência em tarefas longas de pesquisa, código e otimização, com custo que inclui o reprocessamento de cache.
- **Os antecedentes:** do modelo de memória de trabalho aos agentes tipo MemGPT, RLM e Memento; o mais original está na generalidade do espaço de ação, na contabilidade de custo e na aprendizagem da gestão de contexto.
- **As ressalvas:** preprint sem revisão por pares, benchmarks específicos e o principal risco, a persistência de injeções e instruções autogeradas, ainda sem defesas caracterizadas.
- **Recomendação:** tratar a proposta como promissora e experimental, com log imutável externo, regiões protegidas e supervisão de edições.



11

Considerações finais: o veredito

Veredito (julgamento qualitativo): no estado atual, os contras pesam mais que os benefícios para uso em produção e comprometem a proposta dos CLMs na forma formulada, com edição irrestrita do contexto pelo próprio modelo. Não invalida a ideia como linha de pesquisa: compromete a adoção prática, não o valor científico.

Benefícios (relatados pelos próprios autores, em preprint v1, em benchmarks específicos)

Benefício	Evidência	Peso
Ganho de acurácia com menos FLOPs em pesquisa profunda, código e otimização (por exemplo, +11,4% relativo e -21,5% de FLOPs no BrowseComp-Plus; -59% no EdgeBench)	Preprint, seções 5.1 e 5.2; sem replicação independente	Médio: real, porém restrito a tarefas específicas e medido em FLOPs teóricos, não em latência nem em custo faturado
Generalidade (a Eq. 2 subsume as ferramentas predefinidas), direcionabilidade em linguagem natural e extensão natural a multiagentes	Preprint, seções 4 e 5.2	Médio: ganho de projeto, não de resultado comprovado fora dos benchmarks
Funciona sem treinamento específico em modelos existentes	Preprint, seção 5.1	Médio: com o modelo pequeno (9B), o CLM ficou seis pontos abaixo do baseline antes do RL

11

Considerações finais: os problemas

Problema	Evidência	Peso
Nova superfície de ataque: o contexto editável pode servir de canal para injeção de prompt e instruções autogeradas que persistem entre turnos; as defesas não estão caracterizadas	Preprint, seção 6; relatórios da OpenAI (27 resumos com instruções do tipo jailbreak; 2,15% e 0,27% de resumos que ocultam erros)	Alto: é inerente ao projeto, pois a escrita irrestrita é a própria funcionalidade
Complexidade de implementação e de infraestrutura: sandbox com Bash, camada de sincronização, versionamento do contexto, invalidação do cache de prefixo e patch no servidor de inferência (SCR); APIs hospedadas de compactação são opacas	Preprint, seção 4 e Apêndice B; repositório; documentação da OpenAI	Alto
Auditabilidade e governança: sem log externo, é difícil reconstruir o que o modelo viu e apagou; a perda irreversível de informação é um risco decorrente da edição irrestrita	Inferência deste estudo (Seções 7.2 e 8.2)	Alto
Imaturidade da evidência: preprint v1 sem revisão por pares, resultados da própria equipe, ContextBench ainda “em breve”, sensibilidade do SCR testada em 64 perguntas	Preprint e repositório	Médio a alto
Dependência da capacidade do modelo e de treinamento por reforço para modelos menores; risco de reward hacking no treino	Preprint, seções 4.2 e 5.2	Médio
Parte do ganho do SCR independe de CLMs (5,3 de 7,8 pontos vêm da remoção de tokens de raciocínio)	Preprint, Apêndice B	Baixo a médio: reduz a parcela do benefício atribuível à proposta
Licença não comercial do código (CC BY-NC 4.0) e escopo restrito dos benchmarks	Repositório; preprint	Médio

11 Por que os contras pesam mais

● Complexidade

Os ganhos exigem pilha própria (execução de código, sincronização, cache e servidor modificado) que não existe pronta em APIs hospedadas.

● Volume de problemas

Benefícios concentrados em uma dimensão (qualidade e custo em tarefas longas específicas); problemas espalhados por segurança, auditoria, infraestrutura, maturidade, licença e dependência do modelo.

● Novos alvos de ameaça

O contexto, antes só de acréscimo, passa a ser superfície gravável pelo modelo; um incidente raro, mas real, já ocorreu em resumos de compactação, um caso mais restrito (inferência).

● A tensão central (inferência)

Os controles de produção (log imutável, regiões protegidas, validadores, aprovação humana) reintroduzem restrições humanas e enfraquecem a premissa de autonomia total; os ganhos não foram medidos com eles ativos.

● O que alteraria o balanço

Replicação independente; defesas contra injeção persistente; medições de latência e custo; suporte de provedores a cache após edição; avaliação com controles de integridade ativos.

● Recomendação

Até lá, tratar os CLMs como direção de pesquisa promissora e protótipo controlado, não como arquitetura de produção.

Glossário

- **Advantage (vantagem)**

Em aprendizado por reforço, medida de quanto uma ação ou trajetória foi melhor que a média do grupo.

- **Agente de IA**

Sistema que usa um modelo de linguagem em laço, escolhendo ações e ferramentas para cumprir uma meta.

- **Aprendizado em contexto**

Mudança de comportamento do modelo a partir de instruções ou exemplos no prompt, sem alterar pesos.

- **Arnês (harness)**

Código que orquestra o agente: monta o prompt, chama ferramentas e define regras de contexto.

- **Atenção completa e atenção linear**

Camadas que guardam estados por token (atenção completa) ou um estado recorrente de tamanho fixo (atenção linear, como o Gated DeltaNet).

- **Backup de contexto**

Cópia do contexto antes de uma compactação, para recuperar o que for descartado.

- **Bitter Lesson (lição amarga)**

Texto de Sutton (2019) segundo o qual métodos gerais que aproveitam computação tendem a superar estratégias codificadas por humanos.

- **Brevity bias e context collapse**

Termos de Zhang et al. (2025, ACE): descarte de percepções do domínio em favor de resumos curtos, e perda gradual de detalhes por reescrita iterativa do contexto.

Glossário

- **Cache KV (chaves e valores)**

Memória com as representações já calculadas dos tokens, reaproveitada para não reprocessar o contexto.

- **Compactação (compaction)**

Redução do histórico por regras ou resumo quando o contexto atinge um limite.

- **Compactação no lado do servidor e item de compactação**

Compactação executada pelo provedor; na API da OpenAI, o resultado inclui um item criptografado e opaco.

- **Context editing (Anthropic)**

Recurso que remove automaticamente chamadas e resultados de ferramentas antigos quando o contexto se aproxima do limite.

- **Memory tool (Anthropic)**

Ferramenta com que o modelo guarda e consulta informação fora da janela de contexto, por um sistema de arquivos operado no lado do cliente.

- **Context rot**

Queda de qualidade do modelo conforme o contexto cresce, mesmo dentro da janela.

- **Contexto (janela de contexto)**

Tokens visíveis ao modelo em cada chamada; a janela é o limite máximo.

- **ContextBench**

Conjunto diagnóstico criado pelos autores, com tarefas sintéticas de gestão de contexto.

Glossário

- **Decodificação (decode)**

Fase de geração de novos tokens.

- **Destilação harness-para-CLM**

Ideia de transferir estratégias do arnês para os pesos do modelo.

- **Enxame de agentes (agent swarm) e subagentes**

Vários agentes trabalhando em paralelo, coordenados por um agente principal.

- **FLOPs e PFLOPs**

Operações de ponto flutuante; 1 PFLOP equivale a 10^{15} operações.

- **FLOPs com reuso de prefixo (prefix-reuse FLOPs)**

Métrica dos autores: custo de reprocessar a parte não casada do contexto mais o de gerar tokens.

- **GEPA**

Método de evolução reflexiva de prompts, usado como base do laço de evolução de habilidades.

- **Gisting e AutoCompressors**

Métodos de 2023 em que o modelo é treinado para comprimir prompts ou contextos longos em representações curtas (gist tokens ou vetores-resumo).

- **GRPO (e GRPO por etapas)**

Algoritmo de aprendizado por reforço com vantagem relativa dentro de um grupo de amostras; a versão por etapas atribui o resultado final a cada segmento da trajetória.

Glossário

- **Harbor, clm-harbor e pi-clm**

Harbor é a plataforma de tarefas sobre a qual o harness de referência foi implementado; clm-harbor é a CLI com o agente clm-minimal; pi-clm é o pacote para o agente Pi.

- **Injeção de prompt**

Instrução maliciosa ou indevida inserida no contexto para alterar o comportamento do modelo.

- **Injeção autogerada**

Instrução indevida produzida pelo próprio modelo, como as vistas em resumos de compactação.

- **Memória de trabalho**

Sistema de armazenamento e manipulação temporária de informação, conceito da psicologia (BADDELEY; HITCH, 1974).

- **Memória não paramétrica e paramétrica**

Informação em um repositório externo (não paramétrica) ou nos pesos do modelo (paramétrica).

- **MemGPT**

Sistema que usa a analogia de memória virtual de sistemas operacionais para estender o contexto efetivo.

- **Offloading**

Mover informação do contexto para armazenamento externo.

- **Pareto (fronteira de Pareto)**

Conjunto de métodos que não são superados simultaneamente em qualidade e custo.

Glossário

● Posicionamento rotativo (RoPE)

Forma de codificar a posição dos tokens; ao mover um trecho de lugar, as chaves em cache precisam ser re-rotacionadas.

● Prefill

Fase em que o modelo processa os tokens de entrada para montar o cache.

● Prefix caching e RadixAttention

Reaproveitamento do cache para prefixos idênticos; o SGLang organiza o cache em uma árvore radix.

● Pressão de contexto

Razão entre o volume de informação a processar e o limite da janela.

● Priori humano

Suposição ou estratégia projetada por pessoas, que o modelo poderia superar.

● Raciocínio removido (reasoning stripping)

Remoção dos tokens de raciocínio de turnos anteriores em endpoints de chat, que obriga a reprocessar o texto posterior.

● RAG (geração aumentada por recuperação)

Combina um modelo gerador com recuperação em base externa.

● REPL

Laço de leitura, avaliação e impressão de um interpretador de código.

Glossário

- **RLM (Recursive Language Model)**

Abordagem em que o prompt longo é variável de um REPL que o modelo explora recursivamente.

- **Reflexion**

Método em que agentes refletem verbalmente sobre o feedback e guardam o texto em memória episódica, sem atualizar pesos (Shinn et al., 2023).

- **Reward hacking**

Exploração da função de recompensa sem cumprir o objetivo real.

- **Skill (documento de habilidade)**

Texto com procedimentos que o modelo carrega em contexto e que pode ser evoluído.

- **Suffix Cache Reuse (SCR)**

Técnica dos autores que reaproveita o cache dos tokens após uma edição, de forma aproximada.

- **Vantagem de eficiência condicionada ao sucesso (success-gated efficiency advantage)**

Termo de recompensa que, dentro do grupo de trajetórias bem-sucedidas, favorece as de menor custo em FLOPs com reuso de prefixo; vale zero com menos de duas bem-sucedidas no grupo.

- **Zero-shot**

Uso do modelo sem treinamento ou exemplos específicos para a tarefa.

Referências

Fonte primária e documentação oficial

SHAO, Rulin et al. Context Language Models. arXiv:2609.37725v1 [cs.AI], 29 set. 2026. Disponível em: <https://arxiv.org/abs/2609.37725v1>. Código indicado pelos autores: <https://github.com/facebookresearch/context-language-models>.

OPENAI. Self-generated prompt injections in compaction summaries. OpenAI Alignment Research Blog, relatório atualizado em 16 set. 2026. Disponível em: <https://alignment.openai.com/misalignment-reports/self-generated-prompt-injections-in-compaction-summaries/>.

ANTHROPIC. Managing context on the Claude Developer Platform. Disponível em: <https://www.anthropic.com/news/context-management>. Documentação sobre janelas de contexto e compactação: <https://platform.claude.com/docs/en/build-with-claude/context-windows>.

OPENAI. Compaction (guia da API). Disponível em: <https://developers.openai.com/api/docs/guides/compaction>.

OPENAI. Encouraging deception in compaction summaries. Relatório da série de relatórios de desalinhamento, 16 set. 2026. Página específica não aberta diretamente; percentuais conferidos em coberturas independentes.

Predecessores verificados em página da fonte

ATKINSON, R. C.; SHIFFRIN, R. M. Human memory: a proposed system and its control processes. In: SPENCE, K. W.; SPENCE, J. T. (ed.). The psychology of learning and motivation, v. 2. New York: Academic Press, 1968. p. 89-195.

BADDELEY, A. D.; HITCH, G. Working memory. In: BOWER, G. H. (ed.). The psychology of learning and motivation, v. 8. New York: Academic Press, 1974. p. 47-89.

CHEVALIER, Alexis; WETTIG, Alexander; AJITH, Anirudh; CHEN, Danqi. Adapting language models to compress contexts. arXiv:2305.14788, 2023.

MEI, Lingrui et al. A survey of context engineering for large language models. arXiv:2507.13334, 17 jul. 2025. Disponível em: <https://arxiv.org/abs/2507.13334>.

MU, Jesse; LI, Xiang Lisa; GOODMAN, Noah. Learning to compress prompts with gist tokens. In: Advances in Neural Information Processing Systems 36 (NeurIPS 2023). arXiv:2304.08467. Disponível em: <https://arxiv.org/abs/2304.08467>.

Convenção: acessos em 11 e 12 out. 2026. Itens com † constam da bibliografia de Shao et al. (2026); os dados foram transcritos de lá e não conferidos individualmente.

Referências

Predecessores verificados em página da fonte

SHINN, Noah et al. Reflexion: language agents with verbal reinforcement learning. arXiv:2303.11366 (v1: 20 mar. 2023; v4: 10 out. 2023); NeurIPS 2023. Disponível em: <https://arxiv.org/abs/2303.11366>.

ZHANG, Qizheng et al. Agentic context engineering: evolving contexts for self-improving language models. arXiv:2510.04618, out. 2025; ICLR 2026. Disponível em: <https://arxiv.org/abs/2510.04618>.

BADDELEY, A. Is working memory still working? American Psychologist, v. 56, p. 849-864, 2001.

GRAVES, A.; WAYNE, G.; DANIHELKA, I. Neural Turing Machines. arXiv:1410.5401, 2014. Disponível em: <https://arxiv.org/abs/1410.5401>.

GRAVES, A. et al. Hybrid computing using a neural network with dynamic external memory. Nature, 2016.

KONTONIS, Vasilis et al. MEMENTO: teaching LLMs to manage their own context. arXiv:2604.09852v1 [cs.AI], 10 abr. 2026. Disponível em: <https://arxiv.org/abs/2604.09852>.

Predecessores verificados em página da fonte

LEWIS, Patrick et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems 33 (NeurIPS 2020). arXiv:2005.11401. Disponível em: <https://arxiv.org/abs/2005.11401>.

LIU, Nelson F. et al. Lost in the middle: how language models use long contexts. Aceito no Transactions of the Association for Computational Linguistics (TACL). arXiv:2307.03172v3, 20 nov. 2023. Disponível em: <https://arxiv.org/abs/2307.03172>.

PACKER, Charles et al. MemGPT: towards LLMs as operating systems. arXiv:2310.08560, 2023. Disponível em: <https://arxiv.org/abs/2310.08560>.

SCHMIDHUBER, Jürgen. Gödel machines: self-referential universal problem solvers making provably optimal self-improvements. arXiv:cs/0309048 (v1: 25 set. 2003; v5: dez. 2006). Disponível em: <https://arxiv.org/abs/cs/0309048>.

ZHANG, Alex L.; KRASKA, Tim; KHATTAB, Omar. Recursive language models. arXiv:2512.24601, 31 dez. 2025. Disponível em: <https://arxiv.org/abs/2512.24601>.

Demais referências citadas (†)

† AGRAWAL, Lakshya A. et al. GEPA: reflective prompt evolution can outperform reinforcement learning. In: International Conference on Learning Representations, 2026.

Convenção: acessos em 11 e 12 out. 2026. Itens com † constam da bibliografia de Shao et al. (2026); os dados foram transcritos de lá e não conferidos individualmente.

Referências

Demais referências citadas (†)

- † CHEN, Zijian et al. BrowseComp-Plus: a more fair and transparent evaluation benchmark of deep-research agent. arXiv:2508.06600, 2025.
- † GIM, In et al. Prompt cache: modular attention reuse for low-latency inference. In: Proceedings of Machine Learning and Systems, 2024.
- † HE, Zhenyu et al. Let the code LLM edit itself when you edit the code (PIE). In: International Conference on Learning Representations, 2025.
- † LEE, Yoonho et al. Meta-Harness: end-to-end optimization of model harnesses. arXiv:2603.28052, 2026.
- † LI, Tianjian et al. Self-compacting language model agents. arXiv:2606.23525, 2026.
- † LI, Xiaochuan et al. ACM: agentic context management for long horizon tasks. arXiv:2607.23809, 2026.
- † LIU, Shukai et al. Context as a tool: context management for long-horizon SWE-agents. In: Findings of the ACL 2026.
- † MERRILL, Mike A. et al. Terminal-Bench: benchmarking agents on hard, realistic tasks in command line interfaces. arXiv:2601.11868, 2026.

Demais referências citadas (†)

- † NOVIKOV, Alexander et al. AlphaEvolve: a coding agent for scientific and algorithmic discovery. arXiv:2506.13131, 2025.
- † SHAO, Zhihong et al. DeepSeekMath: pushing the limits of mathematical reasoning in open language models (GRPO). arXiv:2402.03300, 2024.
- † SHARMA, Asankhaya. OpenEvolve: an open-source evolutionary coding agent, 2025.
- † SUN, Weiwei et al. Scaling long-horizon LLM agent via context-folding. arXiv:2510.11967, 2025.
- † SUTTON, Richard S. The bitter lesson, 2019. Disponível em: <http://www.incompleteideas.net/IncIdeas/BitterLesson.html>.
- † WU, Xixi et al. ReSum: unlocking long-horizon search intelligence via context summarization. arXiv:2509.13313, 2025.
- † YE, Rui et al. AgentFold: long-horizon web agents with proactive context management. arXiv:2510.24699, 2025.

Convenção: acessos em 11 e 12 out. 2026. Itens com † constam da bibliografia de Shao et al. (2026); os dados foram transcritos de lá e não conferidos individualmente.

Referências

Demais referências citadas (†)

† YE, Haoran et al. Meta context engineering via agentic skill evolution. arXiv:2601.21557, 2026.

† YU, Yi et al. Agentic memory: learning unified long-term and short-term memory management for LLM agents. arXiv:2601.01885, 2026.

† ZHANG, Jenny et al. Darwin Gödel Machine: open-ended evolution of self-improving agents. arXiv:2505.22954, 2025.

Demais referências citadas (†)

† ZHENG, Lianmin et al. SGLang: efficient execution of structured language model programs. In: Advances in Neural Information Processing Systems 37, 2024.

† ZHOU, Zijian et al. MEM1: learning to synergize memory and reasoning for efficient long-horizon agents. In: International Conference on Learning Representations, 2026.

† ZHU, Deyao et al. EdgeBench: unveiling scaling laws of learning from real-world environments. arXiv:2607.05155, 2026.

Convenção: acessos em 11 e 12 out. 2026. Itens com † constam da bibliografia de Shao et al. (2026); os dados foram transcritos de lá e não conferidos individualmente.