Autoreason e AutoResearch

Loops de auto-refinamento que sabem quando parar

Gabriel Rondon

RQ-020 • Research Factory • 15 de abril de 2026

TL;DR

AutoResearch (Karpathy) automatiza pesquisa quando existe uma *metrica* pra otimizar. **Autoreason** (SHL0MS, NousResearch) estende isso pra dominios *subjetivos* (copy, estrategia, texto), substituindo o numero por um torneio adversarial com juizes cegos. Cada iteracao produz tres candidatos - o incumbente A, uma revisao rival B e uma sintese AB - e um painel de agentes frescos elige o vencedor via Borda count. Se A sobrevive duas rodadas seguidas, o loop para. **O ganho depende do gap entre gerar e auto-avaliar:** e maximo em modelos de tiers medios (Haiku 3.5, Gemini Flash) e desaparece nos extremos. Dentro de pipelines de negocio autonomos tipo show-1, Autoreason e o motor das etapas subjetivas (positioning, copy, brand voice, grant writing) enquanto AutoResearch cuida das etapas com metrica (conversao, revenue, pass rate).

1 O problema: por que auto-refinamento ingenuo falha

Pedir pra um LLM "melhore este texto" em loop e uma ideia obvia, e tambem uma ideia que *sistematicamente piora o output*. Tres falhas estruturais explicam isso [3, 4]:

1. **Prompt bias.** Se voce manda o modelo "encontrar problemas", ele encontra - mesmo quando nao ha. E um reflexo sicofantico: o modelo satisfaz a instrucao, nao a realidade. Critica alucinada vira "correcao" que degrada.
2. **Scope creep.** Cada passagem tende a *crescer*: mais bullets, mais qualificacoes, mais secoes. Juizes e autores LLM tem viez de verbosidade [10]. Em 15 passes de "critique & revise", um texto de 500 palavras vira 932 - e o que era claro vira prolixo.
3. **Falta de freio.** Modelos quase nunca dizem "esta bom, nao mexe". A instrucao de revisar sobrepoe qualquer prudencia. Sem mecanismo explicito de parada, o loop corrode o que funcionava.

O resultado pratico: passar de 1 para 15 iteracoes pode *piorar* um texto. SlopCodeBench [9] documenta o mesmo em codigo: agentes degradam progressivamente quando iteram sem verificacao externa.

2 AutoResearch: quando existe um numero

2.1 A ideia de Karpathy

AutoResearch [1] e um loop simples mas poderoso: um agente formula hipoteses, roda experimentos, mede, itera. O criterio de parada e objetivo - `val_bpb`, acuracia, taxa de conversao - e o loop consegue rodar *sem supervisao humana* porque o numero decide.

Funciona onde o mundo ja responde:

- **Otimizacao de modelos.** A perda vai pra baixo ou nao vai.
- **Programacao competitiva.** Os testes passam ou nao passam.
- **Growth metrics.** Conversao sobe ou desce.
- **A/B real.** Trafego divide, a amostra cresce, o vencedor aparece.

> **Intuicao.** AutoResearch transforma o loop de pesquisa num sistema fechado: o mundo (via metrica) e o juiz. O agente so precisa propor experimentos diversos e ler resultados.

2.2 O gap que AutoResearch nao cobre

Mas a maioria das decisoes que importam no dia-a-dia de um produto **nao tem metrica direta**:

- O positioning do lancamento esta certo?
- A copy da landing page pega ou e generica?
- Esse hook de email funciona ou so existe?
- Esse relatorio editorial planta a ideia ou parece dossie de IA?

Nao da pra rodar A/B em positioning estrategico antes de decidir; a amostra e pequena demais, o sinal e lento demais, e as opcoes sao caras demais pra testar todas. E ai que entra Autoreason.

3 Autoreason: o loop que sabe quando parar

3.1 Os quatro papeis

Cada iteracao usa **quatro papeis**, sempre instanciados como *agentes frescos* - sem memoria compartilhada, sem contexto das rodadas anteriores. Isso impede o efeito yes-man, onde um mesmo agente concorda consigo mesmo.

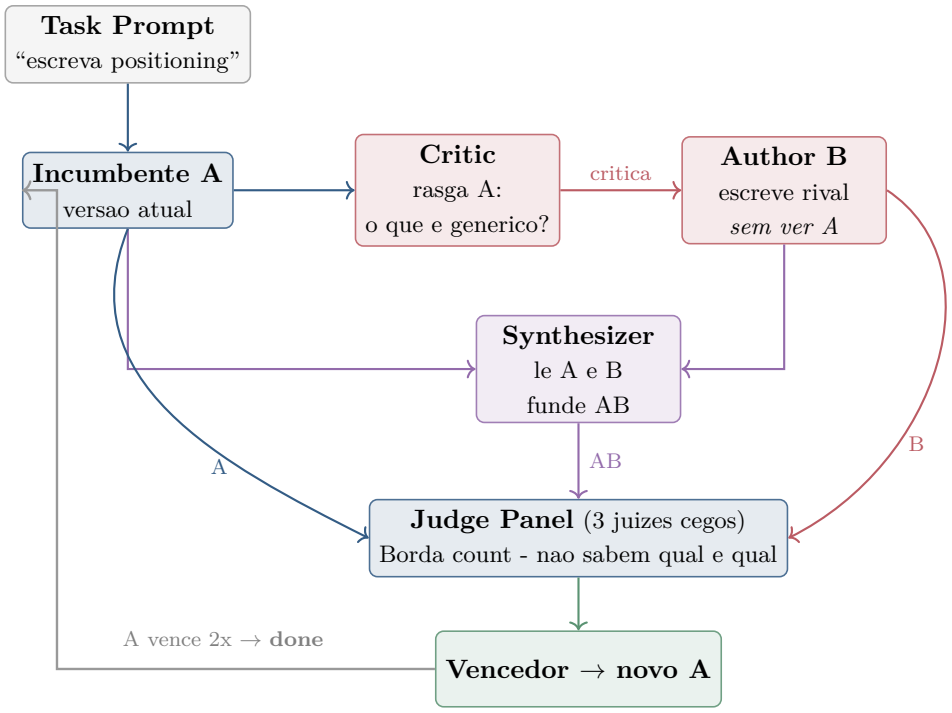

Figura 1. O loop Autoreason. Cada iteracao produz tres candidatos (A, AB, B) avaliados por juizes cegos via Borda count. Se A sobrevive duas rodadas, converge. Todos os papeis sao agentes frescos, sem vazamento de contexto entre eles.

Critic. Le A e destila uma critica afiada: o que soa generico, o que um concorrente poderia dizer palavra por palavra, qual tradeoff nao foi endossado. Nao propoe solucao - so articula o problema.

Author B. Recebe *apenas a critica*, nunca o A original. Escreve um rival do zero. Essa isolacao e central: se visse A, ancoraria; ao nao ver, e forcado a propor algo estruturalmente diferente.

Synthesizer. Le A e B, tenta produzir AB que combina as forcas dos dois. E a voz moderada do torneio.

Judge Panel. Tres juizes *cegos* (nao sabem qual candidato e qual), cada um um agente fresco. Votam via Borda count (ranking ponderado), nao via maioria - Borda e mais robusto a outliers [7].

3.2 Formalizacao

Seja d_t o documento incumbente no passe t. Cada passe produz o conjunto $\{d_t,\ B(d_t),\ S(d_t, B(d_t))\}$ onde B e o operador de revisao adversarial e S e o operador de sintese. Um painel de n juizes atribui rankings $r_i(c)$ ao candidato c. O vencedor e:

$$d_{t+1} = \arg \max_{c \in \{A,\, B,\, AB\}} \sum_{i=1}^{n} (3 - r_i(c))$$

com empates quebrados em favor de d_t. Quando $d_{t+1} = d_t$ por k passes consecutivos (tipicamente $k = 2$), o loop converge.

Por que "do nothing" e de primeira classe. Na maioria dos loops de refinamento, a opcao "nao mudar" nao existe - a cada passe algo precisa mudar. Autoreason torna A competidor legitimo do torneio. Um texto ja bom tende a sobreviver, e dois passes de sobrevivencia e o sinal oficial de parada.

3.3 Ablacoes: por que tres candidatos sao necessarios

O paper testa remover cada componente. Os resultados sao claros:

- **Sem B (sem revisao adversarial):** converge em ∼3 passes. Sem pressao, A vence trivialmente.
- **Sem AB (sem sintese):** converge em ∼2 passes. O torneio colapsa.
- **Com os tres:** ∼24 passes ate estabilizar, com aumento real de qualidade.

A tensao de tres candidatos sustenta o torneio. Dois colapsa.

4 A descoberta central: a curva capacidade-valor

4.1 O gap geracao-avaliacao

Auto-refinamento exige duas capacidades: *gerar* alternativas e *avaliar* qual e melhor. Evidencias empiricas indicam que **avaliar e mais dificil que gerar** [4]. O tamanho desse gap muda com o modelo:

- **Modelo muito fraco** (ex: Llama 8B): nao gera alternativas diversas. O torneio nao tem o que escolher.
- **Modelo forte sweet-spot** (ex: Haiku 3.5, Gemini Flash): gera bem, mas avalia mal. *Autoreason brilha aqui* - os juizes externos (mesmo sendo do mesmo modelo) salvam o modelo de si mesmo.
- **Modelo forte demais** (ex: Sonnet 4.6): ja se auto-avalia razoavelmente bem. Autoreason oferece pouco, e em tarefas muito abertas pode ate piorar.

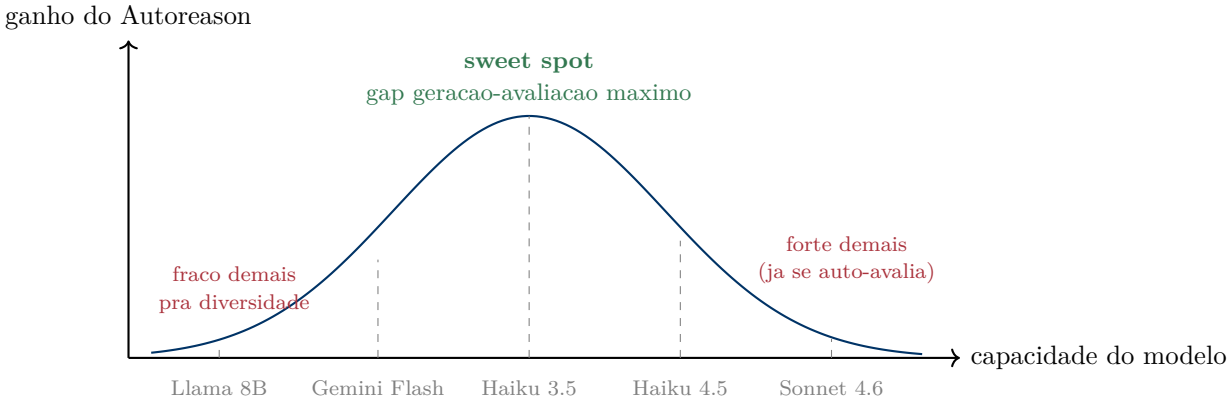

Figura 2. A curva capacidade-valor do Autoreason. O ganho cresce ate o tier medio, onde o gap entre gerar e avaliar e maximo, e decresce conforme o modelo passa a se auto-avaliar bem sozinho. Haiku 4.5 e o ponto de transicao empiricamente observado: mantem ganhos em testes visiveis mas os ganhos em testes privados desaparecem.

4.2 Evidencias numericas

Modelo	Tier	Resultado	Leitura
Llama 8B	base	vence 1/3 tasks	nao gera diversidade pro torneio
Haiku 3.5	**sweet spot**	**42/42 Borda perfeito**	sozinho degrada; com Autoreason lid
Sonnet 4	medio-alto	64% codigo	ganho modesto, consistente
Sonnet 4.6 (task livre)	forte	ultimo lugar (Borda 7)	auto-avaliacao ja suficiente
Sonnet 4.6 (task com escopo)	forte	**1o lugar (Borda 30)**	escopo salva o loop de derivar
Haiku 4.5	transicao	ganho em public-test, nulo em private	gap se fechando

Tabela 1. Autoreason vs. linhas de base, por tier. O sweet-spot aparece em Haiku 3.5; Sonnet 4.6 mostra que escopo muda o jogo.

Implicacao pratica. Autoreason nao e "melhor em tudo". E melhor onde o modelo consegue gerar alternativas legitimas mas nao consegue distinguir bem qual e melhor. Rodar Autoreason com Sonnet 4.6 em texto livre desperdica compute. Rodar Autoreason com Haiku 3.5 em 3 tarefas paralelas pode entregar Borda perfeito a 1/10 do custo.

5 As quatro condicoes pra auto-refinamento funcionar

O paper destila quatro condicoes observadas empiricamente:

Condicao	O que significa na pratica
1. Verificacao externa	Alguem ou algo fora do autor que julgue. Juizes cegos, testes, metrica.
2. Escopo delimitado	Limite de palavras, template, formato fixo. Sem isso, o loop deriva.
3. Raciocinio estruturado	O critic articula *por que* falhou antes da revisao tentar consertar.
4. Espaco de decisao nao-trivial	A tarefa admite abordagens genuinamente diferentes. Template-filling puro nao se beneficia.

Tabela 2. As quatro condicoes empiricas. Quando uma falta, Autoreason performa como baseline ou pior. As quatro juntas explicam todos os casos onde o metodo venceu.

6 A camada de conhecimento: onde Autoreason vira produto

O loop puro debate a partir de *principios gerais* de copy/estrategia. A contribuicao pratica do thread do Shann Holmberg foi mostrar como plugar uma *knowledge layer* que faz o loop debater a partir dos *seus* dados.

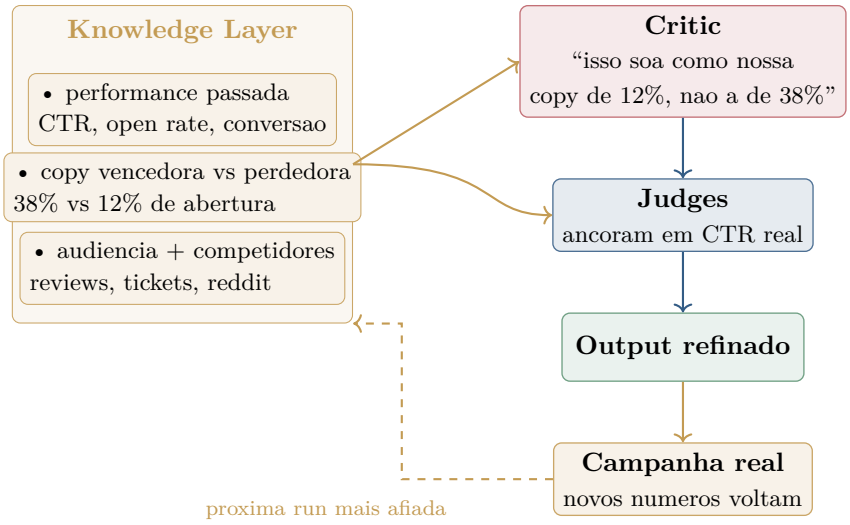

Figura 3. Knowledge layer plugada no Autoreason. Critic e Judges consultam dados historicos; cada novo resultado realimenta a base. O loop nao melhora porque o modelo aprende - melhora porque *o criterio de julgamento* fica cada vez mais proximo dos seus numeros reais.

Exemplo do Shann: rodando em subject lines de email, o Critic passa a dizer “isso se parece com as subject lines que tiveram 12% de abertura, nao com as que tiveram 38%” em vez de argumentar por feeling. Toda a discussao fica ancorada nos dados.

7 Onde Autoreason se encaixa em pipelines de negocio autonomos

Tom Dorr (`@tom_doerr`) e projetos como show-1 (`iamzifei/show-1`) propoem um **sistema operacional de negocios autonomo**: Onboarding → Discover → Strategy → Build → Grow → Operate → Revenue. Esse pipeline e ortogonal a Autoreason - e um *scaffolding* de tarefas. Autoreason e um *motor de raciocinio* que se pluga em cada etapa que produz output subjetivo.

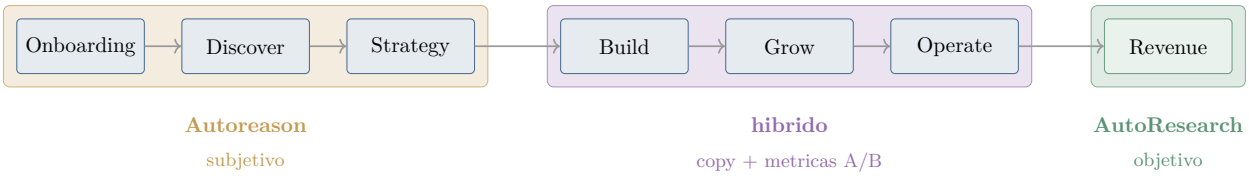

Figura 4. Pipeline estilo show-1 anotado. Etapas subjetivas (positioning, estrategia, SOPs) se beneficiam de Autoreason. Etapas de medicao real (revenue) sao territorio de AutoResearch. Growth e um hibrido: copy e subjetiva mas o A/B fecha o loop objetivo.

8 Aplicacoes nos projetos proprios

Sem forcar - so onde a estrutura bate:

Projeto	Onde pluga	Knowledge layer
DeFarm	Positioning por audiencia (investidor, parceiro de dados, grant SCF)	Pitch decks anteriores, feedback SCF, mensagens aprovadas pelo founder
thepitch.report	Relatorios editoriais de jogadores (texto em 3a pessoa, sem metrica)	Textos aprovados vs rejeitados, estilo The Athletic / O Globo
Filtro de Ouro	Script de qualificacao WhatsApp (copy) + A/B de conversao (metrica)	**Hibrido**: Autoreason gera; AutoResearch escolhe via conversao real
Grants (SCF, OpenSats, Spiral)	Draft de proposta e release notes	Tranches anteriores, criterios publicos, reviews de pares
OSS PR descriptions	Descricao de PR, release notes, postmortem	PRs aceitos vs rejeitados no mesmo repo
Genealogia	**Nao cabe**: verificacao factual, resposta binaria	—

Tabela 3. Mapeamento. Onde o output e texto subjetivo e existe uma base de conhecimento curada, Autoreason e candidato. Onde existe metrica direta, e AutoResearch. Onde a resposta e factual binaria, nao se aplica.

9 Relacao com literatura adjacente

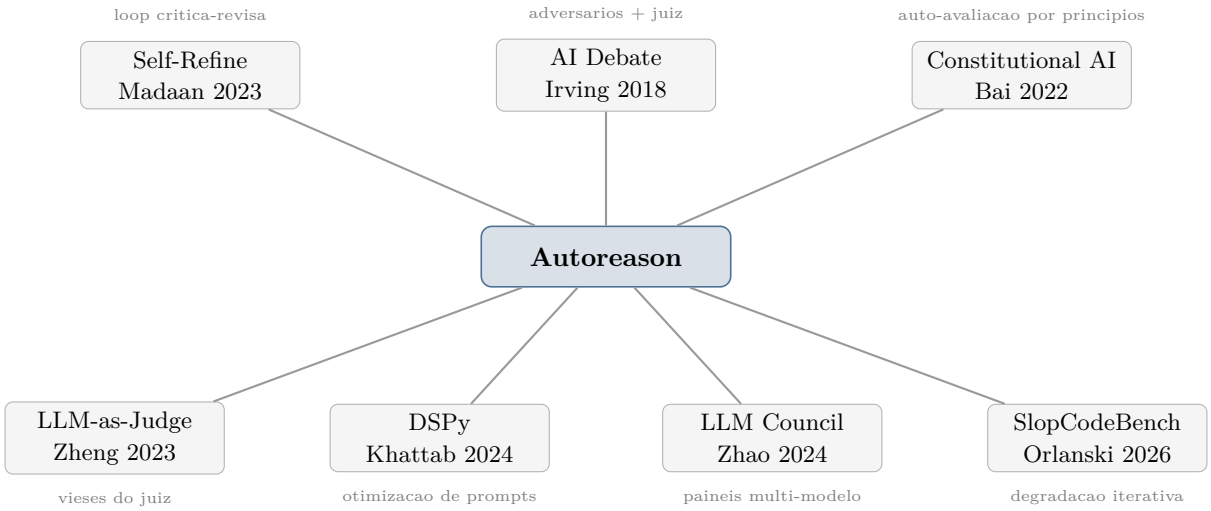

Figura 5. Autoreason na paisagem. Toma de Self-Refine [3] a ideia de critica-revisao; de AI Debate [5] a estrutura adversarial com juiz; de LLM-as-Judge [7] a conciencia de vieses (e os mitiga via agentes frescos); de Constitutional AI [6] a ideia de criterio explicito; de DSPy [8] o uso de feedback estruturado como sinal. SlopCodeBench [9] e a motivacao: documenta empiricamente por que loops ingenuos degradam.

10 Riscos e limitacoes

- **Custo**. Cada passe sao ∼6 chamadas LLM (Critic, Author B, Synthesizer, 3 Judges). Com 10-25 passes ate convergencia, sao 60-150 chamadas por tarefa - ∼6× por passe e ∼10× total vs baseline de um agente so. Compensa quando a qualidade vale mais que o custo de compute, nao antes.

- **Knowledge layer e o trabalho real**. O loop e facil de implementar (algumas centenas de linhas de Python). Curar uma base de conhecimento que faca o Critic falar em termos dos *seus* numeros e o que da o diferencial - e onde todo mundo trava.
- **Avaliacao humana cega pendente no paper**. Quase todos os numeros vem de juizes LLM, nao de humanos. A pasta `human_eval/` do repo contem o material cego mas ainda nao ha resultados publicados. E uma limitacao honesta.
- **Uma familia de modelos**. Todos os experimentos usam Anthropic. Generalizacao pra OpenAI/Google/open-source e hipotese, nao fato.
- **Risco de convergencia prematura**. Sem B ou sem AB, o loop converge em 2-3 passes falsamente. Se alguem implementar uma variante simplificada "pra economizar compute", provavelmente colapsa.
- **Tarefas com escopo aberto e modelo forte**: caso ruim documentado. Sonnet 4.6 em task sem escopo ficou em ultimo. Nao rodar Autoreason nesse cenario.

11 Plano de acao pratico

1. **Experimento piloto**. Rodar `run_overnight.py` do repo `NousResearch/autoreason` contra uma tarefa real nossa. Candidato principal: *positioning DeFarm pra SCF tranche 3*, texto de 500 palavras com escopo.
2. **Avaliacao cega com o founder**. Single-pass (Sonnet 4.6) vs Autoreason (Haiku 3.5) vs 1 prompt humano tuneado. Gabriel classifica sem saber qual e qual.
3. **Se o resultado justificar**: abstrair num CLI simples, por exemplo `autoreason positioning.md -knowledge defarm_kb/`. Plugar nas skills `defarm-biz-market-positioning` e `defarm-biz-pitch-deck`.
4. **Knowledge layer inicial**. Comecar com o minimo: 3 pitches anteriores, 2 feedbacks do SCF, 5 exemplos de copy aprovada vs rejeitada. Crescer organicamente.
5. **Segunda aplicacao candidata**: thepitch.report. Dominio mais subjetivo, sem A/B, knowledge layer e "o que Gabriel aprovou". Teste natural da hipotese.

12 Conclusao

Autoreason nao e um milagre - e um metodo cirurgico que resolve tres falhas estruturais especificas do auto-refinamento ingenuo: prompt bias, scope creep e falta de freio. Faz isso isolando papeis (Critic, Author B, Synthesizer, Judges) em agentes frescos, forcando uma competicao de tres candidatos, e dando a "nao mexer" status de primeira classe.

O ganho real vem de duas escolhas arquiteturais:

1. **Os agentes nao compartilham contexto**. Nao tem como virar eco camera.
2. **Existe criterio de parada**. Se A sobrevive duas rodadas, acabou.

A descoberta cientifica central nao e o loop em si - e a **curva capacidade-valor**. Autoreason nao ajuda o modelo mais fraco (nao tem diversidade), nao ajuda o modelo mais forte (ja sabe se avaliar), e ajuda muito no meio. Essa regiao se desloca pra cima conforme os modelos melhoram: o que era sweet spot em 2026 (Haiku 3.5) ja nao era em 2027 (Haiku 4.5).

Pra nossos projetos, a janela pratica esta aberta agora. Enquanto os modelos medios sao uteis e 10× mais baratos que os frontier, Autoreason e um multiplicador de qualidade aplicavel em positioning, copy editorial, grant writing e descricoes de PR. A infraestrutura tecnica e simples; o valor vira da

curadoria de uma knowledge layer que realmente represente nossos criterios.

Dentro de pipelines maiores (show-1, business OS autonomo), Autoreason ocupa as etapas subjetivas e AutoResearch ocupa as com metrica. Juntos eles fecham a volta: um decide *como* escrever, o outro decide *se* funcionou.

Referências

[1] A. Karpathy. *Autoresearch*, 2026. https://github.com/karpathy/autoresearch.

[2] SHL0MS, Hermes Agent. *Autoreason: Self-Refinement That Knows When to Stop*, NousResearch, 2026. https://github.com/NousResearch/autoreason.

[3] A. Madaan et al. Self-Refine: Iterative refinement with self-feedback. *NeurIPS*, 2023. arXiv:2303.17651.

[4] J. Huang et al. Large language models cannot self-correct reasoning yet without external feedback. 2023. arXiv:2310.01798.

[5] G. Irving, P. Christiano, D. Amodei. AI safety via debate. 2018. arXiv:1805.00899.

[6] Y. Bai et al. Constitutional AI: Harmlessness from AI feedback. 2022. arXiv:2212.08073.

[7] L. Zheng et al. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. *NeurIPS*, 2023. arXiv:2306.05685.

[8] O. Khattab et al. DSPy: Compiling declarative language model calls into self-improving pipelines. *ICLR*, 2024. arXiv:2310.03714.

[9] G. Orlanski et al. SlopCodeBench: Benchmarking how coding agents degrade over iterative tasks. 2026.

[10] R. Saito et al. Verbosity bias in preference labeling by large language models. 2023.

[11] S. Holmberg. Autoreason for marketing: positioning, copy, and landing pages, 2026. X/Twitter thread.

[12] T. Doerr. *show-1: Autonomous AI business operating system*, 2026.