Autoreason and AutoResearch

Self-refinement loops that know when to stop

Gabriel Rondon

RQ-020 • Research Factory • April 15, 2026

TL;DR

AutoResearch (Karpathy) automates research whenever there is a *metric* to optimize. **Autoreason** (SHL0MS, NousResearch) extends the idea to *subjective* domains (copy, strategy, prose), replacing the number with an adversarial tournament judged by blind agents. Each iteration produces three candidates – the incumbent A, a rival revision B, and a synthesis AB – and a fresh-agent panel picks the winner via Borda count. If A survives two rounds, the loop halts. **The payoff depends on the gap between generation and self-evaluation**: it peaks at mid-tier models (Haiku 3.5, Gemini Flash) and vanishes at both extremes. Inside autonomous business pipelines like show-1, Autoreason is the reasoning engine for the subjective steps (positioning, copy, brand voice, grant writing) while AutoResearch owns the metric-bound steps (conversion, revenue, pass rate).

1 The problem: why naive self-refinement fails

Asking an LLM to "make this better" in a loop is an obvious idea and also one that *systematically makes outputs worse*. Three structural failure modes explain why [3, 4]:

1. **Prompt bias.** Tell the model to "find problems" and it will find them – even when there are none. It is sycophancy: the model satisfies the instruction rather than reporting reality. Hallucinated critique becomes "fix" that degrades.
2. **Scope creep.** Each pass tends to *grow*: more bullets, more caveats, more sections. LLM authors and judges both prefer longer outputs regardless of quality [10]. Fifteen passes of "critique & revise" turn a sharp 500-word brief into a 932-word hedge.
3. **No restraint.** Models rarely say "leave it alone." The instruction to revise overrides any caution. Without an explicit stop condition, the loop corrodes what was working.

The practical result: going from 1 to 15 iterations can *reduce* quality. SlopCodeBench [9] documents the same pattern in code: agents degrade progressively when they iterate without external verification.

2 AutoResearch: when a number exists

2.1 Karpathy's idea

AutoResearch [1] is a simple but potent loop: an agent proposes hypotheses, runs experiments, measures, and iterates. The stop criterion is objective – `val_bpb`, accuracy, conversion – and the loop can run *without human supervision* because the number decides.

It works wherever the world already answers:

- **Model optimization.** Loss goes down or it does not.

- **Competitive programming.** Tests pass or fail.
- **Growth metrics.** Conversion rises or drops.
- **Real A/B.** Traffic splits, the sample grows, the winner shows.

Intuition. AutoResearch closes the research loop against reality: the world (via a metric) is the judge. The agent only needs to propose diverse experiments and read results.

2.2 What AutoResearch does not cover

But most day-to-day product decisions **have no direct metric**:

- Is the launch positioning right?
- Does this landing page copy land, or is it generic?
- Does this email hook work, or does it merely exist?
- Does this editorial report plant the idea, or read like AI slop?

You cannot run a meaningful A/B on strategic positioning before you decide; the sample is too small, the signal is too slow, and the options are too expensive to all try. That is where Autoreason fits.

3 Autoreason: the loop that knows when to stop

3.1 The four roles

Every iteration uses **four roles**, always instantiated as *fresh agents* – no shared memory, no context from prior rounds. This design kills the yes-man effect where the same agent agrees with itself.

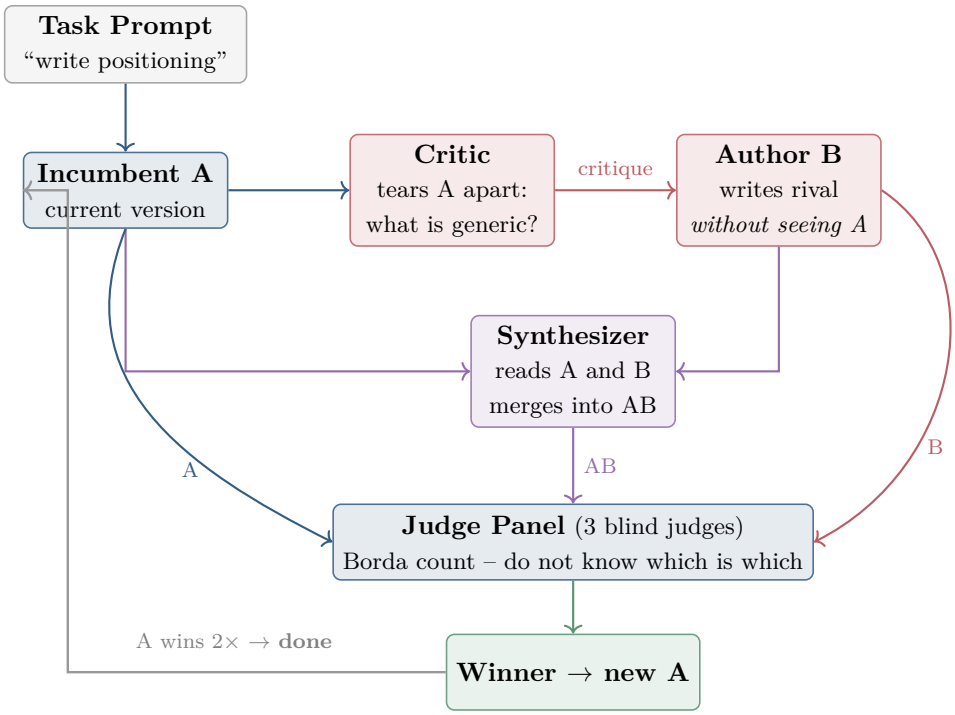

Figure 1. The Autoreason loop. Each iteration produces three candidates (A, AB, B) evaluated by blind judges via Borda count. If A survives two rounds, it converges. All roles are fresh agents, with no context leaking between them.

Critic. Reads A and distills a sharp critique: what sounds generic, what a competitor could say word for word, which tradeoff is unacknowledged. Does not propose a fix – it only articulates the problem.

Author B. Receives *only the critique*, never the original A. Writes a rival from scratch. The isolation is central: if it saw A, it would anchor; without it, it is forced to propose something structurally different.

Synthesizer. Reads A and B, attempts an AB that combines the strengths of both. It is the moderate voice of the tournament.

Judge Panel. Three *blind* judges (they do not know which candidate is which), each a fresh agent. They vote via Borda count (weighted ranking), not by majority – Borda is more robust to outliers [7].

3.2 Formalization

Let d_t be the incumbent document at pass t. Each pass produces the candidate set $\{d_t,\ B(d_t),\ S(d_t, B(d_t))\}$, where B is the adversarial revision operator and S is the synthesis operator. A panel of n judges assigns rankings $r_i(c)$ to candidate c. The winner is:

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

with ties broken in favor of d_t. When $d_{t+1} = d_t$ for k consecutive passes (typically $k = 2$), the loop converges.

Why "do nothing" is first class. In most refinement loops, "leave it alone" is not an option – every pass must change something. Autoreason makes A a legitimate competitor in the tournament. A document that is already good tends to survive, and two consecutive survivals is the official stop signal.

3.3 Ablations: why three candidates are necessary

The paper tests removing each component. Results are clear:

- **Without B (no adversarial revision):** converges in ∼3 passes. With no pressure, A wins trivially.
- **Without AB (no synthesis):** converges in ∼2 passes. The tournament collapses.
- **With all three:** ∼24 passes before stabilizing, with real quality gains.

The tension between three candidates sustains the tournament. Two collapses it.

4 The central finding: the capability-value curve

4.1 The generation-evaluation gap

Self-refinement requires two capabilities: *generating* alternatives and *evaluating* which is best. Empirical evidence suggests **evaluation is harder than generation** [4]. The size of that gap shifts with the model:

- **Too weak** (e.g., Llama 8B): cannot produce diverse alternatives. The tournament has nothing to pick from.
- **Mid-tier sweet spot** (e.g., Haiku 3.5, Gemini Flash): generates well, evaluates poorly. *Autoreason shines here* – external judges (even same-model) save the model from itself.
- **Too strong** (e.g., Sonnet 4.6): self-evaluates reasonably well. Autoreason offers little, and on very open tasks it can even hurt.

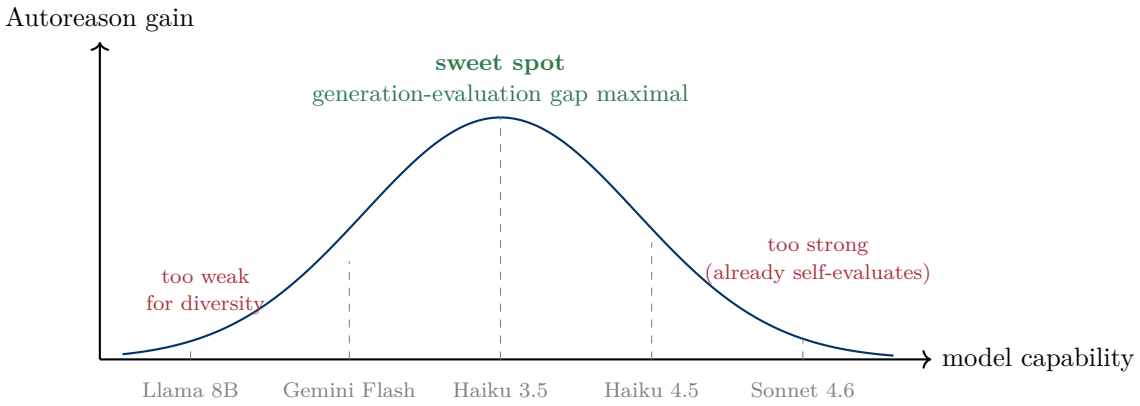

Figure 2. Autoreason's capability-value curve. The gain grows up to the mid-tier, where the gap between generating and evaluating is widest, and decays once the model can self-evaluate well enough on its own. Haiku 4.5 is the empirical transition point: it keeps visible-test gains but loses held-out-test gains.

4.2 Numerical evidence

Model	Tier	Result	Reading
Llama 8B	base	wins 1/3 tasks	no diversity for the tournament
Haiku 3.5	**sweet spot**	**42/42 perfect Borda**	alone it degrades; with Autoreason it leads
Sonnet 4	mid-high	64% code	modest, consistent gain
Sonnet 4.6 (open task)	strong	last place (Borda 7)	self-evaluation already sufficient
Sonnet 4.6 (scoped task)	strong	**1st place (Borda 30)**	scope saves the loop from drifting
Haiku 4.5	transition	public-test gain, null private	gap is closing

Table 1. Autoreason vs. baselines, by tier. The sweet spot appears at Haiku 3.5; Sonnet 4.6 shows scope changes the game.

Practical implication. Autoreason is not "better at everything." It is better where the model can generate legitimate alternatives but cannot reliably pick the best. Running Autoreason with Sonnet 4.6 on open prose wastes compute. Running Autoreason with Haiku 3.5 on three tasks in parallel can deliver perfect Borda at 1/10 the cost.

5 Four conditions for self-refinement to work

The paper distills four empirically observed conditions:

Condition	What it means in practice
1. External verification	Someone or something outside the author judges. Blind judges, tests, metrics.
2. Bounded scope	Word limits, templates, fixed formats. Without this, the loop drifts.
3. Structured reasoning	The critic articulates *why* it failed before the revision tries to fix.
4. Non-trivial decision space	The task admits genuinely different approaches. Pure template-filling sees no benefit.

Table 2. The four empirical conditions. When one is missing, Autoreason performs like baseline or worse. Together they explain every case where the method won.

6 The knowledge layer: where Autoreason becomes a product

The pure loop debates from *general* copy/strategy principles. The practical contribution of Shann Holmberg's thread was showing how to plug in a *knowledge layer* that makes the loop debate from *your* data instead.

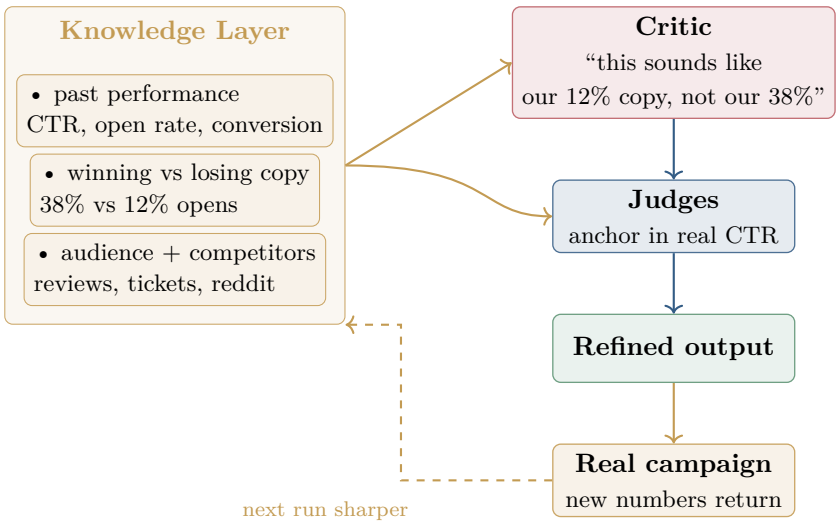

Figure 3. Knowledge layer plugged into Autoreason. Critic and judges query historical data; each new result feeds back into the base. The loop does not improve because the model learns – it improves because *the judgment criterion* gets closer and closer to your actual numbers.

Shann's example: running it on email subject lines, the Critic starts saying "this looks like our 12%-open subject lines, not our 38% ones" instead of arguing from feel. The whole discussion is anchored in data.

7 Where Autoreason fits in autonomous business pipelines

Tom Doerr (`@tom_doerr`) and projects like show-1 (`iamzifei/show-1`) propose an **autonomous business operating system**: Onboarding → Discover → Strategy → Build → Grow → Operate → Revenue. That pipeline is orthogonal to Autoreason – it is a *task scaffolding*. Autoreason is a *reasoning engine* that plugs into each step that emits subjective output.

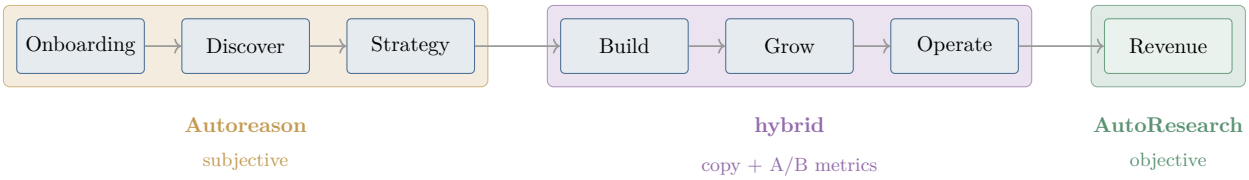

Figure 4. An annotated show-1-style pipeline. Subjective steps (positioning, strategy, SOPs) benefit from Autoreason. Measurement steps (revenue) are AutoResearch territory. Growth is a hybrid: copy is subjective but A/B closes the objective loop.

8 Applications to our own projects

Not forcing it – only where the structure matches:

Project	**Where it plugs in**	**Knowledge layer**
DeFarm	Positioning per audience (investor, data partner, SCF grant)	Past decks, SCF feedback, founder-approved messaging
thepitch.report	Editorial player reports (third-person prose, no metric)	Approved vs rejected texts, The Athletic / O Globo style
Filtro de Ouro	WhatsApp qualification script (copy) + A/B conversion (metric)	**Hybrid**: Autoreason generates; AutoResearch picks via real conversion
Grants (SCF, OpenSats, Spiral)	Proposal drafts and release notes	Past tranches, public criteria, peer reviews
OSS PR descriptions	PR descriptions, release notes, postmortems	Accepted vs rejected PRs in the same repo
Genealogy	**Does not fit**: factual verification, binary answer	—

Table 3. Mapping. Where the output is subjective prose and a curated knowledge base exists, Autoreason is a candidate. Where a direct metric exists, it is AutoResearch. Where the answer is binary factual, it does not apply.

9 Relation to adjacent literature

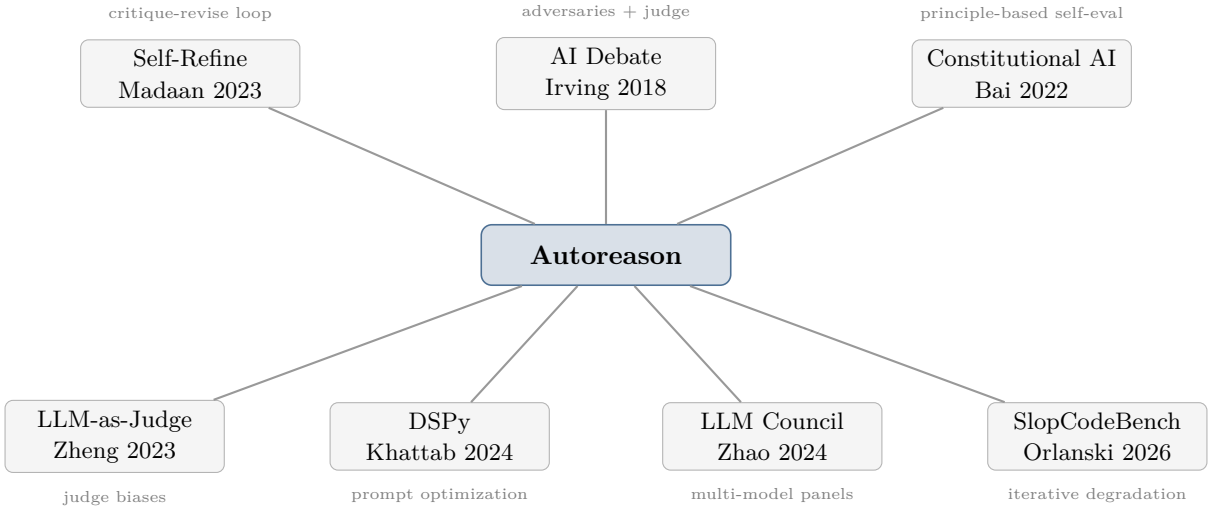

Figure 5. Autoreason in the landscape. It takes from Self-Refine [3] the critique-revise idea; from AI Debate [5] the adversarial-with-judge structure; from LLM-as-Judge [7] the awareness of judge biases (and mitigates them via fresh agents); from Constitutional AI [6] the explicit criterion; from DSPy [8] the use of structured feedback as signal. SlopCodeBench [9] is the motivation: it empirically documents why naive loops degrade.

10 Risks and limitations

- **Cost.** Each pass is ∼6 LLM calls (Critic, Author B, Synthesizer, 3 Judges). With 10–25 passes to converge, that is 60–150 calls per task – ∼6× per-pass and ∼10× total vs a single-agent baseline. It is worth it when quality matters more than compute cost, not before.
- **The knowledge layer is the real work.** The loop is simple to implement (a few hundred lines of Python). Curating a knowledge base that makes the Critic speak in terms of *your* numbers is the differentiator – and where most people get stuck.
- **Blind human evaluation still pending in the paper.** Nearly all numbers come from LLM judges, not humans. The repo's `human_eval/` folder ships blinded materials, but published results are not there yet. An honest limitation.
- **One model family.** All experiments use Anthropic. Generalization to OpenAI/Google/open-source is hypothesis, not fact.
- **Premature-convergence risk.** Without B or without AB, the loop converges in 2–3 passes falsely. Anyone implementing a "simplified" variant to save compute will probably collapse it.
- **Open-scope tasks with strong models**: documented bad case. Sonnet 4.6 on unscoped tasks ranked last. Do not run Autoreason in that setup.

11 Practical action plan

1. **Pilot experiment.** Run `run_overnight.py` from `NousResearch/autoreason` against a real task of ours. Primary candidate: *DeFarm positioning for SCF tranche 3*, a 500-word brief with explicit scope.

2. **Blind evaluation with the founder.** Single-pass (Sonnet 4.6) vs Autoreason (Haiku 3.5) vs a hand-tuned human prompt. Gabriel ranks without knowing which is which.
3. **If results justify it**, abstract into a simple CLI, e.g. `autoreason positioning.md -knowledge defarm_kb/`. Plug into skills `defarm-biz-market-positioning` and `defarm-biz-pitch-deck`.
4. **Minimum-viable knowledge layer.** Start small: 3 past pitches, 2 SCF feedbacks, 5 examples of approved vs rejected copy. Grow organically.
5. **Second application candidate**: thepitch.report. More subjective domain, no A/B, and the knowledge layer is "what Gabriel approved." A natural test of the hypothesis.

12 Conclusion

Autoreason is not a miracle – it is a surgical method that addresses three specific structural failures of naive self-refinement: prompt bias, scope creep, and lack of restraint. It does so by isolating roles (Critic, Author B, Synthesizer, Judges) in fresh agents, forcing a three-candidate competition, and giving "do nothing" first-class status.

Real gains come from two architectural choices:

1. **Agents do not share context**. There is no way to become an echo chamber.
2. **An explicit stop criterion exists**. If A survives two rounds, it is done.

The central scientific finding is not the loop itself – it is the **capability-value curve**. Autoreason does not help the weakest model (no diversity), does not help the strongest model (already self-evaluates), and helps substantially in between. That region shifts upward as models improve: 2026's sweet spot (Haiku 3.5) is no longer 2027's (Haiku 4.5).

For our own projects, the practical window is open now. While mid-tier models are useful and $\sim 10\times$ cheaper than frontier ones, Autoreason is a quality multiplier applicable to positioning, editorial copy, grant writing, and PR descriptions. The technical infrastructure is simple; the value comes from curating a knowledge layer that actually encodes our criteria.

Inside larger pipelines (show-1, autonomous business OS), Autoreason owns the subjective steps while AutoResearch owns the metric-bound ones. Together they close the loop: one decides *how* to write, the other decides *whether* it worked.

References

[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.