

Self-Refine Without a Repetition Penalty

A controlled case study in recursive model collapse, and the one setting that fixed it

Ben Rees¹

¹ www.bjrees.com, ben@bjrees.com

Abstract

This is the empirical companion to the Cyberdyne local/cloud routing framework set out on 10 August 2026, and to the shorter blog account of the same events. That framework predicted that recursive self-refine on a local model would widen breadth without closing a reasoning gap. This paper reports what actually happened when I tested it: two full-budget runs of the identical task on the identical model, six and a half hours each, one with no guardrails and one with two specific fixes applied. The first collapsed. The second didn't. The gap between them is smaller, and more informative, than the original hypothesis expected.

The question: does recursion buy a small model anything?

Self-Refine is a real, published technique: generate an output, have the same model critique it, revise using that critique, repeat. No extra training, no second model. Madaan et al. tested it across seven tasks on GPT-3.5, ChatGPT, and GPT-4, and reported an average improvement of roughly 20% over single-pass generation (Madaan et al., 2023). It is not a fringe idea. It is a documented, working method for extracting more from a model at inference time rather than training time.

The question this paper asks is narrower and more practical: does it work the same way on a much smaller model, run for much longer, with no supervision? I have a local machine, a Mac Mini running a 35-billion-parameter open-weight model (Qwen3.6-35B, ternary-quantized to roughly 9.4GB via TurboQuant-MLX) at around 10 tokens per second, and hours of spare overnight compute I am not paying cloud API rates for. The Cyberdyne routing framework's working hypothesis was that this combination, a weak model with abundant free time, could trade wall-clock time for the single-pass reasoning quality it does not have, at least on breadth-shaped tasks: generating many candidate ideas rather than one correct answer. That hypothesis was stated in advance, in writing, before either experiment ran, specifically so it could be falsified rather than fitted after the fact.

Experiment one: the same loop, no guardrails

The task, held constant across both runs, was to generate fifteen to twenty distinct content-angle ideas for two categories of search query that were, at the time, returning zero AI-search visibility for my own site. Each cycle: generate a list, critique it in a separate call, produce a revised list, feed that list into the next cycle as the thing to critique again. No external judge, no cloud model in the loop, no retrieval. The run was capped at 6.5 hours of wall-clock time and, as a backstop only, 300 cycles.

The generation and refinement calls used nucleus sampling (temperature 0.8, top-p 0.95, top-k 20) with no repetition penalty configured, a default the underlying generation script simply did not set. The run completed 179 cycles before hitting the time cap.

What a hundred and seventy-nine cycles actually produced

Cycle 1, a single generate call with no critique step yet, took 97 seconds and produced a clean, coherent list: fifteen distinct ideas, no repeated phrasing. It was also, by every measure applied afterwards, the best output the run produced. Nothing after it was clearly better.

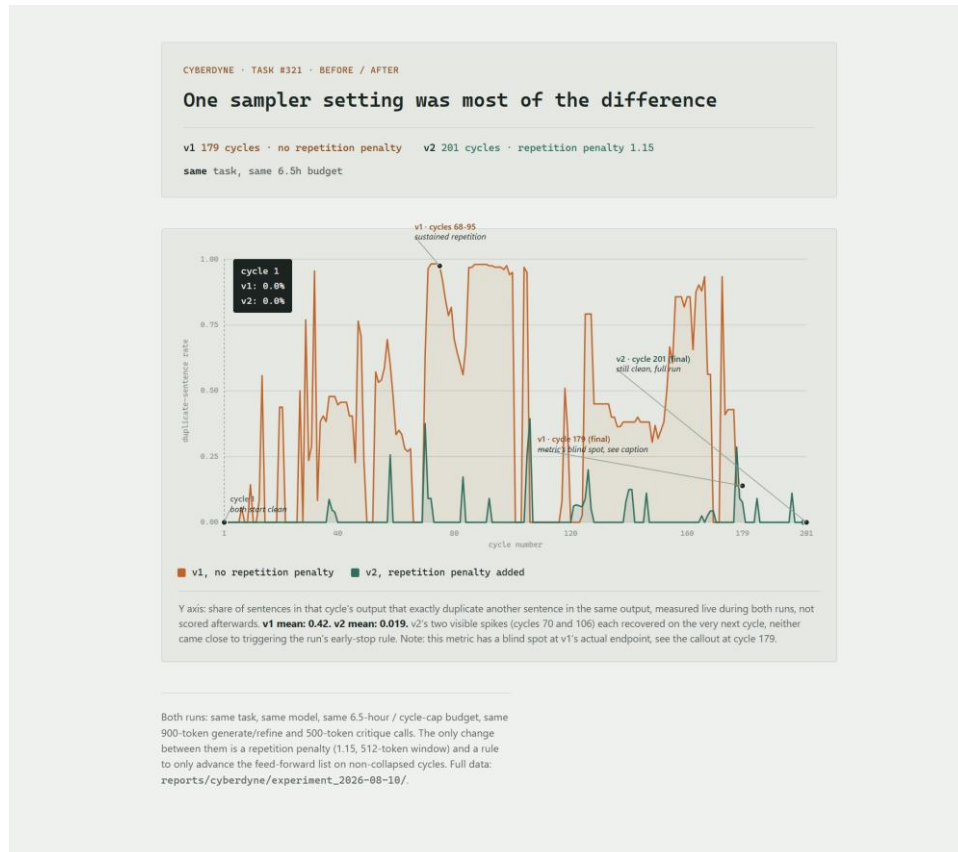


Figure 1. Cycle 1: both runs begin identically, at a repetition score of 0.0%. Whatever divergence follows is entirely a product of what happens next, not a different starting point.

By cycles 20 to 40, list items began repeating verbatim in blocks, the same five ideas recurring as items six through ten, then again as eleven through fifteen. Between roughly cycles 68 and 95, the run entered a sustained high-repetition plateau, holding at 0.92 to 0.98 on a live duplicate-sentence-rate measure (the fraction of sentences in an output that exactly duplicate an earlier sentence in the same output) for nearly thirty consecutive cycles. By cycle 80 specifically, one list item repeated the fragment “It cannot be defended” eight times in succession before the token budget cut it off.

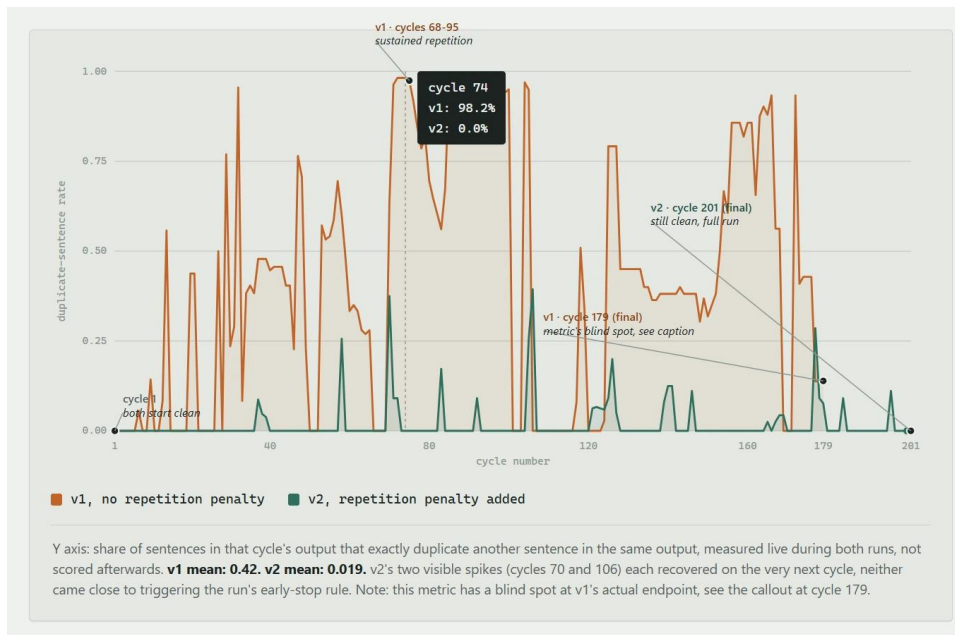


Figure 2. Cycle 74, inside the plateau: experiment one reads 98.2%, close to its ceiling on this measure. Experiment two, for reference, is still at 0.0% at the same cycle number.

By cycle 179, the final cycle, the model had stopped attempting the brief altogether and was reciting its own system instructions back as prose: *“I will generate a list of 15-20 distinct content angle ideas. I will ensure each one is a specific, opinionated hook...”*

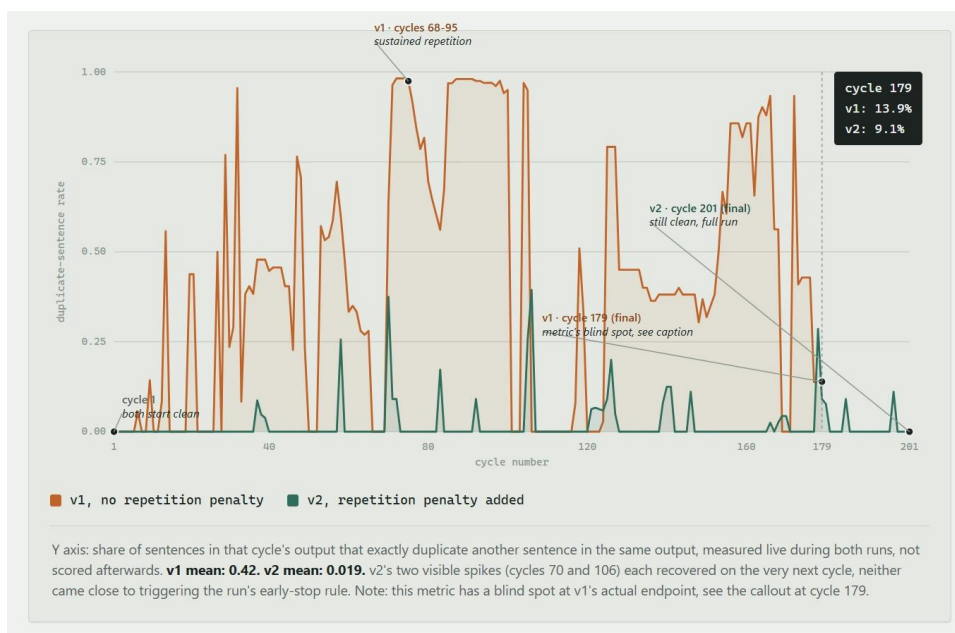


Figure 3. Cycle 179, experiment one's endpoint: 13.9% on the chart, considerably lower than the plateau at cycle 74. Read as a number this looks like recovery. Read as text, discussed below, it is the worst output of the run.

That final cycle's raw repetition score, 0.14, is lower than the mid-run plateau's 0.95, which understates rather than overstates how degenerate it was: the measure only catches exact repeated sentences, and by cycle 179 the model was varying single words inside an otherwise fixed template rather than repeating whole sentences outright, a failure mode a purely lexical metric is blind to. Read as text rather than as a number, cycle 179 is the worst output of the run. The mean repetition score across all 179 cycles was 0.42.

Two mechanisms, not one

Two configuration choices, not a fact about model size, are the most likely explanation, and both point to the same underlying literature.

First, no repetition penalty was active. Holtzman et al. (2020) showed that likelihood-based decoding in neural language models tends toward bland, repetitive text unless the sampling process actively discourages it, which is the motivation for nucleus sampling in the first place. Nucleus sampling was configured here. A repetition penalty, a separate and complementary mechanism that down-weights tokens already used recently, was not, and nothing in the pipeline supplied one by default: the model's own `generation_config.json` carries no repetition-penalty value to inherit.

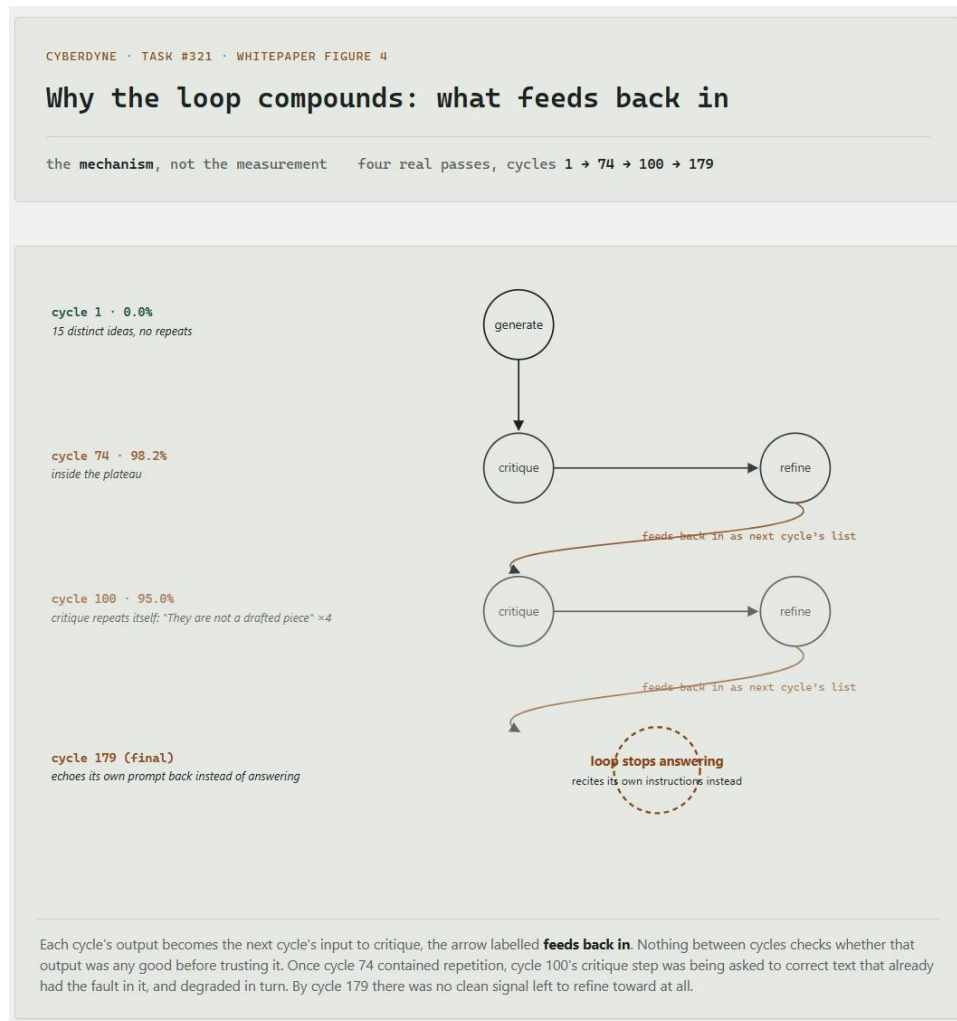


Figure 4. Each cycle's output becomes the next cycle's input to critique, the arrow labelled "feeds back in." Nothing between cycles checks whether that output was any good before trusting it, which is why the fault introduced by cycle 74 is still present, uncorrected, by cycle 179.

Second, and compounding the first, every cycle fed the previous cycle's raw output back in as the literal text to critique and revise. Once a cycle contained a repetition loop, the next cycle was being asked to critique and improve text that already exhibited the failure. The critique step itself was not immune: by cycle 100, the critique output repeated the phrase "They are not a drafted piece" four times in a row, meaning the signal meant to correct the drift was itself degrading before it reached the refine step.

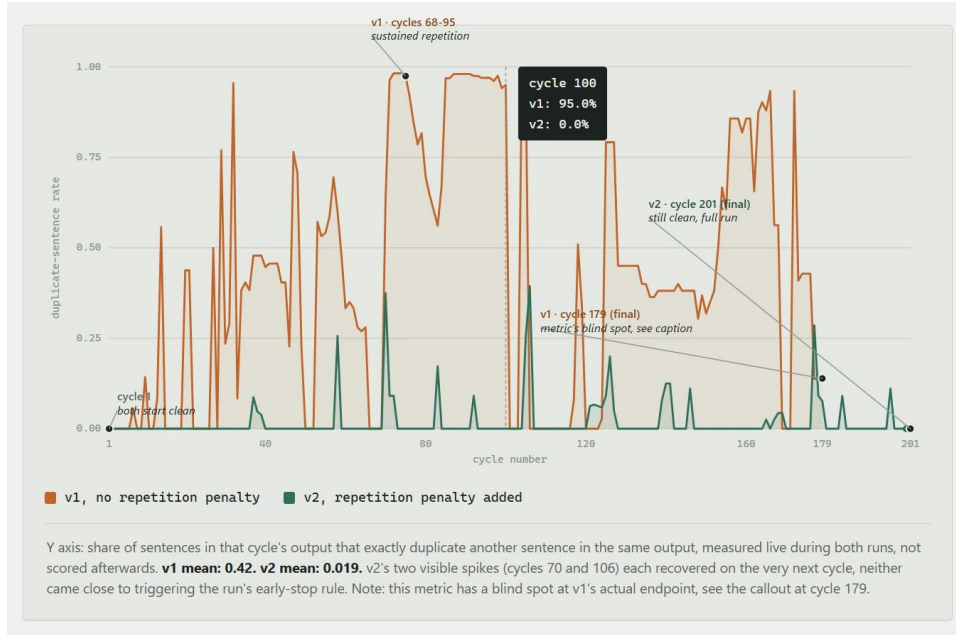


Figure 5. Cycle 100: 95.0% on experiment one, one of the run's most severe individual readings, and the specific cycle where the critique output itself started repeating rather than correcting.

This is close in spirit to the family of repetition metrics introduced by Welleck et al. (2020), whose seq-rep-n measures the proportion of duplicated n-grams in generated text; the duplicate-sentence-rate metric used throughout this paper is a coarser, sentence-level cousin of that idea, chosen for legibility when spot-checking specific transcripts rather than as a claim of methodological novelty.

Neither of these is a statement about what a 35-billion-parameter model is capable of in principle. Both are statements about what happens when a specific, correctable omission compounds over 179 unsupervised iterations.

Experiment two: the same loop, with both fixes

Two changes, and only two, were made before rerunning the identical task on the identical model with the identical 6.5-hour budget. First, a repetition penalty of 1.15, applied over a 512-token context window, added to every generation call. Second, the loop no longer advances to a cycle whose output scores as collapsed (duplicate-sentence rate at or above 0.35): it retries the next cycle from the last known-good state instead, with a hard stop after six consecutive collapsed cycles. Everything else, the prompt, the task, the sampling temperature, the token budgets per call, the wall-clock and cycle caps, was held constant.

Two hundred and one cycles, and it held

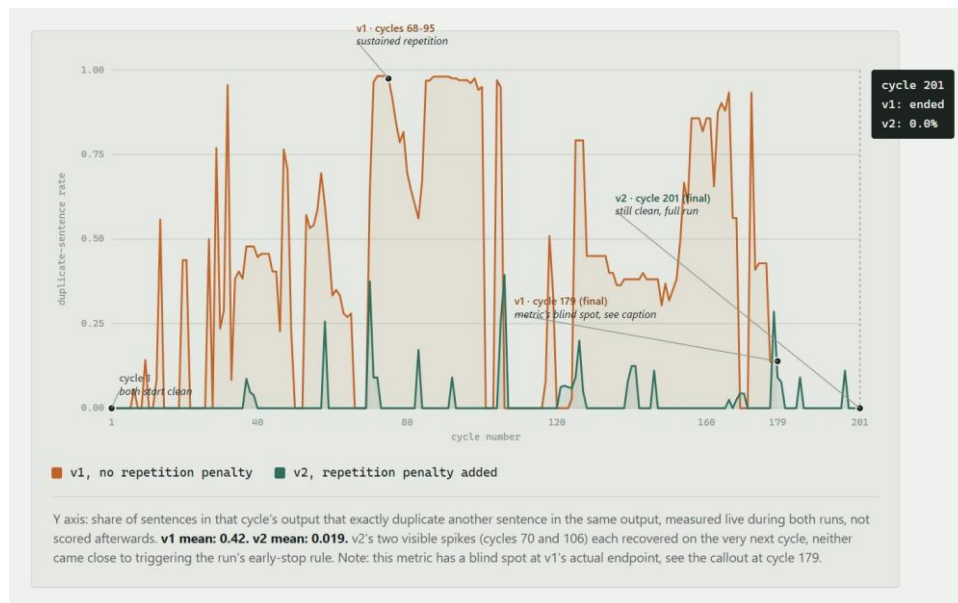


Figure 6. Cycle 201, experiment two's endpoint: still 0.0%, the full run completed with no repetition and no early-stop. Experiment one, for reference, ended 22 cycles earlier, in the state shown in Figure 3.

The second run completed 201 cycles inside the 6.5-hour budget and never triggered the early-stop rule. Of those 201 cycles, exactly two (cycle 70 and cycle 106) crossed the 0.35 collapse threshold, and both recovered fully on the very next cycle without the retry mechanism needing to intervene beyond that single step. The mean duplicate-sentence rate across the entire run was 0.019, against 0.42 for the unguarded run. A spot check of the final cycle's output found it still substantive and on-topic, not perfect prose (a couple of run-on sentences appear), but nothing resembling the collapse that ended experiment one.

The uncomfortable finding: one fix did the work of two

The two fixes were expected to matter roughly comparably. They did not. With only two brief excursions above threshold across 201 cycles, and both self-correcting without the retry-and-gate mechanism doing anything beyond the single retry, the repetition penalty alone accounts for nearly all of the improvement. The more elaborate mechanism, tracking output quality live and refusing to build on a degraded state, functioned as intended on the two occasions it was needed, but it was needed twice in 201 cycles, not the load-bearing safeguard it was designed to be.

This matters for how the finding should be read. It would be easy, and wrong, to conclude that recursive self-refine on a small local model is a solved problem requiring careful state management. The more accurate and more useful conclusion is narrower: this specific run collapsed because of one missing, cheap, well-documented setting, and adding it back fixed nearly everything. The elaborate half of the fix earns its place as a safety net, not as the explanation.

What this does and doesn't tell you about local compute

Stability is not the same claim as quality. Two hundred and one cycles free of exact repetition is evidence that the loop did not break; it is not evidence that cycle 201 is worth reading, or that it says anything a single cloud-model pass on the same brief would not have said better, in seconds rather than hours. That comparison, breadth quality against one cloud pass, was outside the scope of this experiment and remains open. The Cyberdyne routing framework's caution stands: local compute is triage for cost and privacy, not a substitute for reasoning quality, and nothing here changes that.

What this does establish is more specific and more actionable. A local self-refine loop that appears to fail is not automatically evidence that the underlying approach or the underlying hardware is inadequate. It may be evidence that a single, checkable configuration default was never set. Before concluding that a small model cannot sustain multi-hour unsupervised iteration, the boring question, is a repetition penalty configured, is worth asking first.

What to actually do with a recursive local loop

Set a repetition penalty explicitly. Do not assume a model's default generation configuration supplies one; check the file, and if it is absent, set one deliberately rather than omitting it by default.

Never feed a cycle's raw output forward unconditionally. Score it, however crudely, before trusting it as the basis for the next iteration. A live check that only advances on non-degraded output is cheap to build and, on the evidence here, cheap insurance rather than the primary fix, but it is the difference between a run that stalls once and recovers and a run that never recovers at all.

Treat “it ran for hours without visibly breaking” and “it produced something worth using” as two separate questions, and do not let evidence for the first stand in for the second. This paper answers the first question with real data. It does not answer the second, and says so plainly rather than implying otherwise.

Related reading

- [A 35-billion-parameter model now runs on a \\$599 Mac Mini](#)
- [I ran a local AI model through 179 rounds of self-refine. It collapsed. One sampler setting fixed it.](#)
- [Your brand doesn't get its own space inside an AI model](#)

References

- Madaan, A., Tandon, N., Gupta, P., et al. (2023). *Self-Refine: Iterative Refinement with Self-Feedback*. NeurIPS 2023. arXiv:2303.17651.
- Holtzman, A., Buys, J., Du, L., Forbes, M., Choi, Y. (2020). *The Curious Case of Neural Text Degeneration*. ICLR 2020. arXiv:1904.09751.
- Welleck, S., Kulikov, I., Roller, S., Dinan, E., Cho, K., Weston, J. (2020). *Neural Text Generation with Unlikelihood Training*. ICLR 2020. arXiv:1908.04319.