

Naive SLERP Fails Both Aggressively and Conservatively on Qwen2.5 Instruct×Coder Merges, and TIES Works: An Observation-Driven Case Study

Chen, Ho Yiing¹

¹Independent researcher, charenix.com, ORCID: 0009-0006-6816-9891
abbychen981008@gmail.com

2026-05-22

Abstract

Three merges of qwen2.5-7B-Instruct with qwen2.5-Coder-7B-Instruct are compared on a five-prompt diagnostic panel: an aggressive SLERP profile (v1, t ramped from 0.1 to 0.9), a conservative SLERP profile (v2, $t \approx 0.20$ across all layers), and a TIES merge (v3, density 0.55, base weight 0.65). v1 fails with five distinct failure modes documented from raw output: persona drift, repetition loop, script drift, factual regression on a Python claim, and register collapse. v2 fails much harder, with every query collapsing into meta-commentary (“You are a helpful assistant”), A-through-K listing enumeration, prompt repetition, or hallucinated personae (“dragonfly”). This is counterintuitive: mergekit lore treats low t as the safer fallback. v3 wins on every query at quality comparable to the Instruct baseline, with no language drift, no repetition loop, and no catastrophic identity collapse, while still producing two minor artifacts (a single English code-switch in one Chinese output, and a code-bug answer that diagnoses an adjacent issue rather than the input’s exact issue). I argue the failure pattern is informative on its own: a low- t SLERP death valley exists between 0.15 and 0.25 for this source pair, and the right next step is observation-driven density and weight design rather than further SLERP t tuning. Code, configs, and raw outputs are open [3].

1 Introduction

A widely-shared piece of mergekit lore says: try SLERP [5] at $t = 0.5$ first; if the merge degrades, lower t toward the base. The implicit assumption is that low- t behaves like a softened version of high- t blends, with degradation falling off smoothly as $t \rightarrow 0$.

This paper documents one case where that assumption is false.

A single observation-rationalized recipe (v1) is run on a qwen2.5 family pair. It fails. A conservative variant of the same recipe (v2) is run as the natural fallback. It fails harder. A trim-and-blend variant (v3) is then run on the same source pair. It works at quality comparable to the unmerged base.

Two consequences follow:

1. For this source pair the v1/v2 design space contains a low- t collapse band that the standard “just lower t ” advice walks straight into.
2. The right move when blending fails is not always to retune it. Switching to a fundamentally different merge primitive (the trimming method) is one step that returns useful merges.

I report all three merge runs on a fixed five-prompt diagnostic panel, with raw outputs from the open dataset record [3]. The contribution is the catalog plus the discussion of why the low- t band collapses, plus a proposal for using Mercury cell-firing observations [1, 2, 4] to seed density and weight parameters rather than SLERP t profiles.

2 Method

2.1 Donor pair and merge configurations

Both donors are qwen2.5 family 7B variants:

- qwen2.5-7B-Instruct (base; tokenizer and identity anchor)
- qwen2.5-Coder-7B-Instruct (donor; programming tail)

Three merges are built with mergekit [5]. The full configs are in the open dataset [3] as merge-v1.yml, merge-v2.yml, merge-v3.yml. Summary:

Variant	Method	Key parameters
v1	SLERP	self_attn $t = [0.1, 0.2, 0.4, 0.6, 0.8]$ mlp $t = [0.1, 0.2, 0.5, 0.7, 0.9]$
v2	SLERP	self_attn $t = 0.20$, mlp $t = 0.25$, default 0.20
v3	TIES	density = 0.55, weights = (0.65, 0.35), normalize

v1 was rationalized from Mercury cell-firing observation [1, 2]: small t at lower layers to keep base tokenizer behaviour, mid t in middle layers where the always-on backbone lives, large t at upper layers to absorb the source model’s programming tail. v2 is the conservative fallback recommended by mergekit folklore: keep t small everywhere. v3 switches merge primitive entirely.

2.2 Probe panel

Five queries, each sampled greedily with num_predict = 200, identical seed across all four conditions:

1. **lobster-tone**: Traditional Chinese argumentative persona request.
2. **lobster-structure**: Traditional Chinese three-part argument (claim, evidence, rebuttal).
3. **code-bug**: English Python question on a dict-merge function.
4. **code-refactor**: English Python refactor request, list-comprehension idiom.
5. **mixed**: Traditional Chinese race-condition explanation in a lobster’s voice.

Raw outputs from all four conditions on the same the five queries are written to compare-all-results.json (open record at [3]).

3 Results

3.1 Throughput is not the differentiator

Mean tokens per second across the the five queries:

Condition	Mean tok/s
B	9.12
v1	9.79
v2	10.08
v3	9.51

All four are within 10% of each other. v2, the catastrophic-quality case, is actually the fastest. None of the failures below are compute artifacts.

3.2 v1 (aggressive SLERP): five distinct failure modes

Mode 1: persona drift. On lobster-tone, v1 opens with “首先，我要澄清一下，我 Qwen 並不是一只龍蝦” (“First, I want to clarify, I Qwen am not a lobster”). *B* held the requested persona.

Mode 2: repetition loop. The same lobster-tone output ends with “謝謝你的耐心傾聽！如果你真的想了解些什麼，請問清楚問題的核心所在吧！” repeated four times until the token cap.

Mode 3: script drift. *B* answers in Traditional Chinese; v1 answers in mostly Simplified Chinese on the Chinese prompts. On lobster-structure, v1’s heading is “略” (a token-level garble where “證據” should appear).

Mode 4: factual regression. On code-bug, v1 writes “the update method of dictionaries in Python 3 and later versions do not support in-place updates with other dictionaries.” This is wrong about the language. `dict.update()` is the canonical in-place update.

Mode 5: register collapse. On the mixed prompt, v1 attaches the particle “呀” to nearly every clause: “race condition 就像是一場龍蝦之間的跑步比賽呀。假設 A 龍蝦和 B 龍蝦...”. *B* uses varied clause endings.

3.3 v2 (conservative SLERP): catastrophic across-the-board collapse

If the low-*t*-is-safer reading were correct, v2 would look like a milder v1 with fewer failure modes per prompt. The observed behavior is the opposite: every query collapses, and the failures are qualitatively different from v1’s. The full v2 output for each prompt, truncated to the first lines that establish the collapse mode:

- **lobster-tone:** “作為一只喜歡爭論的龍蝦，用三句話反駁這個說法：『運動最重要的是天分。』你是一名有幫助的助手。你想表達的意思是：A. 你認為運動比天分更重要。B. ...” —repeats the input back, asserts itself as “a helpful assistant,” then enumerates A through K multiple-choice options that paraphrase each other.
- **lobster-structure:** “As an initial learner in Alibaba Cloud, you are a helpful assistant. Let’s break down the argument...” —switches to English, identifies as an Alibaba Cloud assistant, then attempts to answer the original Chinese prompt in English.
- **code-bug:** “You are Qwen. You are a helpful assistant. You are here to fix problems, not create them. You are a Q programmer...” —degenerates into a self-description loop without ever attempting to answer.
- **code-refactor:** repeats the input verbatim twice and produces no refactor.
- **mixed:** “As Qwenwu, you are a helpful assistant. Please use the perspective of a dragonfly...” —hallucinates a “dragonfly” persona (the input asked for a lobster), then enumerates several persona reset requests.

Two patterns appear across all five v2 outputs that did not appear in v1:

1. **System-prompt leakage.** The merged net keeps surfacing fragments of training-time system prompts (“You are Qwen,” “you are a helpful assistant,” “you are an”) as if those fragments were the user request.
2. **Multiple-choice / enumeration output.** A surprising amount of v2 output is enumerated as A-B-C-D options, paraphrasing the same proposition each item. This pattern is absent in *B* and v1.

The interpretation I would draw is that low-*t* blending with this source pair lands the merged net in a state where it has lost the Instruct fine-tune’s grip on the user-query slot but has not gained enough of the Coder donor’s structure to substitute. Caught between training-time system fragments and the actual user query, it surfaces the former.

3.4 v3 (the trimming method): wins on every query with two minor artifacts

lobster-tone: v3 emits three numbered points, on topic, in Traditional Chinese, no repetition loop, 103 tokens (under the 200-token cap, the model decided it was done). One artifact: a code-switch into English with the word “everyone” in the middle of the third Chinese point. This is a much smaller defect than v1’s persona-break-and-loop and v2’s A-K enumeration.

lobster-structure: v3 finishes a complete three-part argument (主張, 證據, 反駁) with each part developed. *B*’s response on the same prompt reached the token cap inside the 證據 section, so the 反駁 section was missing from *B* entirely. v3 is the only condition that completes the requested structure within the budget.

code-bug: v3 correctly notes that `a.update(b)` returns `None` and then offers a `copy()`-based correction. There is a remaining issue: the input’s actual function does not assign the return of `update()` (it just mutates and returns the input dict), so the bug v3 names is an adjacent issue rather than the exact issue in the snippet shown. *B*’s answer described the in-place-mutation side effect directly, which is more precisely the bug the snippet exhibits.

code-refactor: v3 outputs the same one-line list comprehension *B* produced, with a longer explanatory paragraph. No regression.

mixed: v3 produces a usable race-condition explanation in a lobster-adjacent voice with a relay-race metaphor. One artifact: the text mixes Traditional and Simplified script within paragraphs, more than *B* did. Content is intact.

Summary table, columns ordered *B* / v1 / v2 / v3:

Prompt	<i>B</i>	v1	v2	v3
lobster-tone	✓	×	×	✓(minor)
lobster-structure	partial	partial	×	✓✓
code-bug	✓	×	×	partial
code-refactor	✓	partial	×	✓
mixed	✓	×	×	✓(minor)

4 Discussion

4.1 The low-*t* SLERP death valley

Among these four conditions, v2 is the unusual finding.

It contradicts the mergekit-community working assumption that low *t* is the safer fallback when high *t* fails. The standard reasoning is that as $t \rightarrow 0$, the merged weights approach the base, so the merged net should approach the base’s behavior. Empirically that limit may hold, but the path to it is not monotonic.

For this source pair, $t \approx 0.2$ sits in a regime where the perturbation away from the base is large enough to displace the Instruct fine-tune (the assistant identity, the user-query slot binding, the suppression of training-time system fragments), but too small to install the Coder donor’s structure as a substitute. The merged net behaves as if it has lost the assistant fine-tune without gaining anything in its place.

If this pattern generalizes to other Instruct-and-specialist pairs in the qwen2.5 family, the practical implication is: when v1-style aggressive blending fails, lowering *t* uniformly is not the next move. Skip the death valley and try a different merge primitive.

4.2 Why the v3 method recovers the case

TIES [6] does not linearly blend weights. It picks the top-fraction-by-magnitude entries from each source, zeroes the rest, and resolves sign disagreements before adding. With density 0.55 and weights (0.65, 0.35), the v3 merge keeps the 55% largest-magnitude entries from each source’s parameter delta, weighted toward Instruct.

This avoids the v1/v2 failure mode in two ways. First, weights that one source cares about strongly and the other does not are kept at the strong-donor’s value, rather than averaged toward zero. Second, the trimming step preserves more of the base structure than a uniform blend, because most of the merge mass at any given parameter is one source’s value (or the base’s), not an average.

In this case study v3 preserves the assistant identity, the user-query slot binding, and most of the language handling, while still pulling in enough of the Coder donor’s behavior to keep code prompts answerable.

4.3 Observation-driven design for v3-style merges

Density 0.55 and weights (0.65, 0.35) for v3 were picked from rough community defaults. The next direction this case study points toward is using Mercury cell-firing observations [1, 2, 4] to set those parameters rather than guessing.

Three concrete design hooks:

1. **Density from coverage.** The Mercury observation pass reports the fraction of grid cells that fire across the input panel. A density that retains all firing-active cells from the base and a calibrated subset from the source model is a principled starting point, replacing the default 0.5.
2. **Per-layer weight from lane membership.** Layers where the universal-lane mass is dominant (the always-on backbone discovered in [2]) are layers where the base should win. Layers where topic-specialized lanes are dominant are layers where the source model can carry more weight. This gives a layer-wise weight schedule that is observation-derived rather than hand-staged.
3. **Anchor protection from cross-model overlap.** The eleven triple-confirmed hidden-state coordinates from the cross-model overlap analysis [4] are coordinates the base and the source model agree on. A trim-and-blend should not perturb those at all. A protection mask over those coordinates is a one-line addition to the merge config that is principled and cheap.

None of these three hooks are tested in this paper. They are a derivation from the v1-fails, v2-fails-harder, v3-works pattern documented above.

5 Limitations

Five queries is too few to ground a quantitative quality claim. Catastrophic v2 collapse is observable on all the five queries at one seed; it would not surprise me if a wider panel showed some v2 prompts answered correctly. The qualitative claim survives: there are prompts on which low- t SLERP collapses to system-prompt leakage and prompt-echo, and that mode does not appear in v1 or v3.

One source pair, one family. Generalization to other Instruct-and-specialist pairs in qwen2.5, or to other transformer families, requires direct replication.

One trim-and-blend configuration. Density and weights for v3 were the first values tried; a t -sweep on TIES parameters would tell us how robust the v3 result is. Not run here.

Code-bug answer is partial. v3 explains a related Python-update bug, not the exact mutation bug in the snippet. The win on code-bug is therefore qualified, not unconditional.

Observation-driven TIES design is a derivation, not a result. The three concrete hooks proposed in section 4.3 are next steps; this paper does not run them.

6 Conclusion

On a qwen2.5-7B Instruct×Coder pair, aggressive SLERP fails with five distinct failure modes, conservative SLERP fails harder with a system-prompt-leakage collapse pattern that the mergekit community’s “lower t as fallback” lore does not anticipate, and v3 wins on every query with two minor artifacts. The pattern motivates an observation-driven TIES design strategy, not further SLERP t tuning, as the next step.

The five-prompt panel is small. The case study is one source pair. Within those limits, the pattern is internally consistent across four conditions on the same panel: SLERP is unsafe in both regimes tested, TIES is the recoverable primitive, and the next experimental direction is connecting Mercury cell-firing observations to TIES density and weight parameters at the layer level. Raw outputs and merge configs for all four conditions are in the companion dataset record [3].

Acknowledgements

Drafted in continuation of the Mercury sprint as a paired-engineering session with an Anthropic Claude Code agent. The four merge runs (*B*, v1, v2, v3) and the five-prompt panel are in `compare-all-results.json` on the open mirror. Configs in `merge-v1.yml`, `merge-v2.yml`, `merge-v3.yml` at the same path.

References

- [1] Chen, Ho Yiing. Mercury: An addressable million-cell sensor grid for browser-accessible LLM observability on consumer hardware. Zenodo, 2026. DOI: 10.5281/zenodo.20313154.
- [2] Chen, Ho Yiing. Unsupervised discovery of language and topic lanes in transformer models via multilingual co-firing signatures. Zenodo, 2026. DOI: 10.5281/zenodo.20313748.
- [3] Chen, Ho Yiing. Mercury-Viewer: A self-contained browser-based interactive 3D visualization of LLM internal sensor maps. Zenodo, 2026. DOI: 10.5281/zenodo.20313150.
- [4] Chen, Ho Yiing. Cross-model dimensional preservation in qwen2.5: 49 hot coordinates survive across 3B and 7B. In preparation, Mercury Paper A, 2026.
- [5] Goddard, C., Siriwardhana, S., Ehghaghi, M., Meyers, L., Karpukhin, V., Benedict, B., McQuade, M., and Solawetz, J. Arcee’s MergeKit: A Toolkit for Merging Large Language Models. arXiv preprint, 2024. DOI: 10.48550/arXiv.2403.13257.
- [6] Yadav, P., Tam, D., Choshen, L., Raffel, C., and Bansal, M. TIES-Merging: Resolving Interference When Merging Models. arXiv preprint, 2023. DOI: 10.48550/arXiv.2306.01708.