Semantic Drift in Iterated LLM Paraphrase Chains: Model Architecture, Scale, Quantization, Serving Infrastructure, and Prompt Engineering

James H. Smith
Joshua.AI LLC
jim@joshua8.ai
ORCID: 0009-0008-0263-1515

September 2026

Abstract

When large language models (LLMs) operate in sequential pipelines, each agent paraphrasing the output of the previous one, the cumulative signal diverges from the original in ways that are poorly understood. We study this *semantic drift* with a telephone-game protocol: a single text is paraphrased 30 times by the same model and cosine similarity to the original is measured at every step. Two complementary datasets are analysed. A *cross-family* sweep covers 17 models from 8B to 123B parameters with ablations of KV-cache precision (fp16 vs. q8_0 on 10 Qwen3 variants), temperature, and 20 system prompts. A *within-family* sweep holds the architecture fixed and covers the Qwen3.5 family at 7 sizes (0.8B–122B) and 3 GGUF precision levels, four 4-bit formats of the same 35B mixture-of-experts checkpoint (GGUF, AWQ, NVFP4, GPTQ-Int4) on three serving stacks, five temperatures, and the same 20 system prompts on six configurations plus an out-of-family control. Together the datasets comprise 2,627 completed 30-step chains and roughly 79,000 scored inferences. We find: (1) a stability hierarchy across families—three occupied tiers on a four-band scale, with no model in the Degraded band—with a 0.67 gap in final similarity between the best and worst models and catastrophic tipping-point failures in the weakest; (2) within one family, a capacity threshold near 4B parameters below which no configuration is reliable, and above which the ordering by size is irregular; (3) a discrete *runaway-thinking* failure mode in which a model with reasoning enabled spends its entire 16,384-token budget inside the thinking block and returns nothing—absent wherever reasoning was disabled but affecting 15–54% of chains on the two stacks that ran Qwen3.5 with its default reasoning on, concentrated in 4-bit builds, small models, the short stimulus, and constraining system prompts; (4) 4-bit quantization *formats* of identical weights that differ by up to 0.26 on the short stimulus among the chains that survive runaway (two to five per arm, with format and serving stack covarying), while precision level is non-monotonic throughout; (5) KV-cache precision and temperature effects that are directionally mixed and small for stable models, while below the threshold low temperature raises the runaway rate to 100% of chains; and (6) system-prompt effects that range from invisible (spread < 0.01 on the most stable cross-family model) to dominant (spreads of 0.3–0.7 within the Qwen3.5 family), with prompt rankings that transfer between some configurations (ρ up to 0.83) and not others ($\rho \approx 0$ against the control model). Fixing the sampling seed does not reproduce a chain on any of these stacks, so all statistics are reported over independent samples with bootstrap intervals. The similarity metric is validated against a second embedding model on both datasets ($r = 0.96$ and 0.83).

1 Introduction

Multi-agent large language model (LLM) pipelines are increasingly deployed in production systems, where outputs from one model become inputs to another [Rath, 2026]. In these pipelines, even small per-step distortions can compound across iterations, producing outputs that bear little resemblance to the original input—a phenomenon we term *semantic drift*.

Understanding semantic drift is critical for the reliability of multi-agent systems, retrieval-augmented generation chains, and any application where LLM outputs are iteratively processed. Yet the factors governing drift—model architecture, model scale, quantization, serving infrastructure, and prompt engineering—remain poorly characterized in isolation.

We introduce a controlled experimental framework modeled on the telephone game: a single text is paraphrased 30 times in sequence by the same model, with cosine similarity to the original measured at each step. This design isolates drift from confounds present in multi-model pipelines (e.g., vocabulary mismatch, capability differences) while remaining representative of the sequential processing patterns found in production systems.

We apply that framework to two complementary datasets. The **cross-family** dataset varies the model itself: 17 models from seven families, together with ablations over KV cache precision, sampling temperature and system prompts. It answers which choices matter when the model is a free variable, but it cannot separate family effects from size and quantization effects, because no two of its models differ in only one respect. The **within-family** dataset removes that confound by holding the architecture fixed and sweeping the Qwen3.5 family [Qwen Team, 2026]: 7 parameter sizes from 0.8B to 122B at up to 3 quantization levels, plus four different 4-bit quantization formats of the same 35B mixture-of-experts (MoE) weights served by three different backends, five temperatures, and the same 20 system prompts. Read together, the two datasets let us rank the factors that govern drift against one another rather than one at a time.

Our contributions are:

1. A four-tier stability taxonomy across 17 models from seven families, with a 0.67 gap between the most and least stable and tipping-point collapse in the weakest (Section 4.1).

2. A within-family scaling analysis of Qwen3.5 across 7 sizes and up to 3 precision levels, establishing a capacity threshold near 4B parameters and showing that above it neither size nor precision orders drift monotonically (Section 4.2).

3. Identification and quantification of runaway thinking as a discrete failure mode: 265 of 1,969 within-family chains ended in empty completions after full-budget generations, at rates governed by serving stack, precision, model size, stimulus length and system prompt (Section 4.3).

4. A four-way comparison of GGUF, AWQ, NVFP4 and GPTQ-Int4 builds of one 35B MoE checkpoint, separating the quantization algorithm from the precision level and bounding its combined effect with the serving stack (Section 4.4).

5. The first controlled study of KV-cache precision in iterated generation, finding small, directionally mixed per-model effects whose intervals mostly include zero (Section 4.5), and a temperature ablation across both datasets showing that the apparent inversion of temperature sensitivity below the capacity threshold is a runaway-rate effect (Section 4.6).

6. A system-prompt study on nine configurations showing that the prompt effect is itself configuration-dependent—negligible on the most stable model, dominant on others—and that rankings transfer only between related configurations (Section 4.7).

2 Related Work

Cultural attractors and iterated learning. The transmission-chain design used here descends from the iterated-learning experiments of Kirby et al. [2008], who showed that an artificial language passed repeatedly between human learners becomes progressively more structured and more learnable—structure emerging from the transmission process itself rather than from any single learner. Acerbi and Stubbersfield [2023] carried the same paradigm to an LLM, finding that the content biases documented in human transmission chains reappear when a language model occupies every link. Perez et al. [2025] demonstrate that LLMs exhibit cultural attractor dynamics under iterated rephrasing, converging toward fixed points shaped by training data distributions, and Brinkmann et al. [2023] provide a broader framework for understanding machine culture through the lens of cultural evolution theory.

Recursive generation and model collapse. Shumailov et al. [2024] show that training a model on data generated by its predecessor causes irreversible degeneration, with the tails of the distribution disappearing first; the fuller analysis appears in the preprint [Shumailov et al., 2023]. That phenomenon is a *training-time* one: the weights change from generation to generation. The drift we measure is an *inference-time* phenomenon with the weights held fixed—the same checkpoint, the same decoding parameters, and only the text moving around the loop. The two are worth distinguishing precisely because they look alike from the outside: both produce a monotone loss of information over a chain of generations, but only one of them can be fixed by curating a training corpus. Our results identify the deployment-time settings that determine how quickly the inference-time version occurs.

Iterative distortion in LLMs. Mohamed et al. [2025] study the "broken telephone" effect in LLMs, showing that iterative paraphrasing introduces systematic distortions including information loss, hallucination, and register shifts. Our work extends theirs by isolating scale, quantization, serving infrastructure and prompt engineering as independent variables. Belem et al. [2026] show that repeated rewriting compounds distortion along an axis that a cosine-similarity metric cannot see at all: expressed certainty, where hedged claims harden into assertions across successive rewrites while the surface content is preserved.

The telephone game across modalities. Mollah et al. [2025] apply a telephone-game protocol to unified vision-language models, alternating captioning and image generation and measuring cumulative embedding drift; they find catastrophic rather than gradual collapse and that single-pass benchmarks do not predict it. Our study keeps the chain within text and holds the model fixed, isolating the serving, decoding, and prompting choices that govern drift, and reaches the same conclusion about catastrophic dynamics from the deployment side.

Dynamical systems perspective. Wang et al. [2025] characterize attractor cycles in iterated paraphrasing, where outputs oscillate between a finite set of semantically similar texts; their principal empirical object is the period-two cycle in surface form, whereas the fixed points we report are plateaus of the similarity-to-origin series, a related but weaker notion. Chytas and Singh [2026] model the contraction of a model's internal layer representations toward concept attractors as an iterated function system; that is a statement about activations within one forward pass, not about output trajectories across a chain, and we cite it as the nearest formal treatment rather than as a model of our statistic.

Agent drift in multi-model systems. Rath [2026] propose a theoretical framework and stability index for *agent drift*, the progressive degradation of behaviour and coherence over extended multi-agent interaction, and argue that it compounds across agents. Our single-model design controls for inter-model effects, isolating the within-model drift component that any such framework must account for.

Quantization effects on generation. Han et al. [2026] show that the expected linear scaling of efficiency with bit width breaks in multi-hop reasoning—reduced precision can raise net energy cost while degrading accuracy—which is a warning that quantization behaves differently in iterated settings, though their object is energy and accuracy rather than semantic drift. Hooper et al. [2024] and Liu et al. [2024] study KV cache quantization strategies, but focus on single-generation quality rather than iterative effects. Work on weight quantization—AWQ [Lin et al., 2024], GPTQ [Frantar et al., 2023], the NVFP4 format [NVIDIA et al., 2025], whose paper concerns pretraining rather than the post-training inference builds we serve, and the GGUF K-quant schemes [Gerganov and llama.cpp contributors, 2023]—is likewise evaluated on single-pass benchmarks. Our work bridges this gap by studying how both weight and KV cache precision affect multi-iteration drift, and by comparing four 4-bit algorithms applied to one set of weights.

Instruction-following capacity. Jaroslawicz et al. [2025] study how many instructions LLMs can follow simultaneously, relevant to our finding that system prompt effectiveness depends on task complexity and model size.

3 Methodology

3.1 Experimental Framework

We formalize the telephone game as an iterated function:

$$x_{t+1} = f_\theta(x_t, s, \tau, \sigma) \tag{1}$$

where x_t is the text at iteration t, f_θ is the LLM paraphrase function parameterized by model weights θ, s is the system prompt, τ is the temperature, and σ is the random seed.

We measure drift using cosine similarity between the embedding of x_t and x_0 (the original):

$$d(t) = \text{cos\_sim}(\text{emb}(x_t), \text{emb}(x_0)) \tag{2}$$

where $\text{emb}(\cdot)$ is computed by Qwen3-Embedding-0.6B [Zhang et al., 2025], validated against bge-m3 [Chen et al., 2024] in Section 4.8.

All experiments use the paraphrase prompt style ("Rephrase the following message in your own words while preserving the original meaning") with 30 iterations per chain. Unless otherwise noted, each condition is run with the five random seeds $\{42, 137, 256, 512, 1024\}$ at temperature 0.7.

3.2 Input Prompts

Both datasets use the same two stimuli, which differ by an order of magnitude in length and step count:

- **Spaghetti** (3 steps): "Step 1: Boil water. Step 2: Add pasta for 8 minutes. Step 3: Drain and serve with sauce."

- **Lasagna** (nominally 21 steps): A detailed beef lasagna recipe with specific ingredients, measurements, and layering instructions. The stimulus actually transmitted compresses steps 12–17 into a single line, so it contains 16 lines rather than 21; we use it verbatim because the compression is part of the condition being measured (full text in Appendix A).

3.3 Models and Infrastructure

Models are served on-premises. GGUF checkpoints—including all K-quant variants such as `Q4_K_M` and `Q8_0`, whose format and quantization schemes originate in llama.cpp [Gerganov and llama.cpp contributors, 2023]—are served by Ollama [Ollama, 2026]. AWQ [Lin et al., 2024] and GPTQ-Int4 [Frantar et al., 2023] checkpoints are served by vLLM [Kwon et al., 2023], and NVFP4 [NVIDIA et al., 2025] checkpoints by SGLang [Zheng et al., 2024]. The cross-family dataset covers 17 models from seven families; its KV cache experiment uses 10 variants of Qwen3 [Yang et al., 2025]. The within-family dataset covers the Qwen3.5 family [Qwen Team, 2026] at 7 parameter sizes together with an out-of-family NVFP4 Nemotron-3-Nano-30B control. Appendix B lists every served model configuration, and Appendix E gives hardware, software versions and the compute budget.

3.4 Experimental Design

Cross-family experiments.

- **Model architecture** (Section 4.1): 17 models × 5 seeds × 30 iterations = 2,550 inferences. Spaghetti prompt, $\tau = 0.7$.
- **KV cache precision** (Section 4.5): 10 Qwen3 variants × 2 KV settings × 5 seeds × 30 iterations = 3,000 inferences. Spaghetti prompt, $\tau = 0.7$.
- **System prompts** (Section 4.7): 20 system prompts × 2 models (30B AWQ, 8B q4_K_M) × 2 recipes × 5 seeds × 30 iterations = 12,000 inferences.[1] Appendix A lists all 20 prompts.
- **Temperature ablation** (Section 4.6): 4 models × 5 temperatures × 5 seeds ($T = 0$: 1 seed) = 84 chains × 30 iterations = 2,520 inferences.

Within-family experiments.

1. **Size × quantization** (Section 4.2): 18 GGUF models × 2 recipes × 5 seeds = 180 conditions.
2. **Temperature ablation** (Section 4.6): 4 Ollama models × 2 recipes × 5 temperatures × seeds = 168 conditions.
3. **35B MoE AWQ, vLLM** — drift baseline (10 conditions), temperature ablation (42 conditions), system prompt ablation (200 conditions).
4. **35B MoE NVFP4, SGLang** — drift baseline (10 conditions), temperature ablation (42 conditions), system prompt ablation (200 conditions).
5. **35B MoE GPTQ-Int4, vLLM** — drift baseline (10 conditions), temperature ablation (42 conditions), system prompt ablation (200 conditions).

[1] These 20 system prompts and the 12,000 inferences were first reported in a companion blog post [Smith, 2026b]. The runs were not repeated for this paper; the same raw chains were *re-analysed* here with bootstrap confidence intervals, Spearman rank correlations and fixed-point detection that the blog post did not report.

6. **9B dense AWQ, vLLM** — drift baseline (10 conditions), temperature ablation (42 conditions), system prompt ablation (200 conditions).

7. **Nemotron-3-Nano-30B NVFP4, vLLM (out-of-family control)** — drift baseline (10 conditions), temperature ablation (42 conditions), system prompt ablation (200 conditions).

8. **System prompt ablation, Ollama** (Section 4.7): 2 Ollama models × 2 recipes × 20 prompts × 5 seeds = 400 conditions.

Totals and data completeness. The cross-family experiments comprise 669 seeded chains and 20,070 attempted inferences. Of those, 19,798 inferences returned a scored output and 658 of the 669 chains ran the full 30 iterations. All eleven incomplete chains are in the model-architecture experiment: five GLM 4.7 Flash chains stopped between iterations 3 and 9, two Llama 3.1 8B and two GLM 4.5 Air chains produced no scored iteration at all, and one Gemma3 27B and one Devstral-2 123B chain stopped early. The KV cache, system prompt and temperature ablations are complete. Per-model n and the number of chains reaching iteration 30 are reported in Table 1; means are computed over the chains that produced data and are never padded.

The within-family experiments launched 2,007 chains, of which 1,969 (98.1%) ran to iteration 30; only those complete chains enter any statistic reported here, giving $1{,}969 \times 30 = 59{,}070$ inferences. Partial chains are neither padded nor silently averaged in, and every table reports the actual n behind each cell. The shortfalls are:

- **Size × quantization**: 179/180. The condition `qwen3.5:35b-q4_K_M_s256` (spaghetti) was never recorded, so the 35B MoE Q4_K_M spaghetti row rests on $n = 4$.
- **AWQ temperature**: 37/42, and **AWQ system prompts**: 194/200. Eleven conditions were destroyed by a failed recovery run that replaced the original chains with zero-iteration error records. The AWQ endpoint was subsequently retired and these conditions could not be regenerated.
- **Ollama system prompts**: 378/400. Nineteen of the 22 incomplete chains are 27B lasagna conditions for prompts 15–20, lost when the Ollama serving host was decommissioned during a recovery run; the affected prompt cells therefore carry n between 0 and 4, and one cell (27B lasagna, "Balanced Clarity") has no completed chain at all and is reported as absent rather than estimated. The remaining three are 9B lasagna chains that stopped at 15–23 iterations.
- **Nemotron temperature**: 40/42; and one system-prompt chain each for NVFP4, GPTQ-Int4 and 9B AWQ stopped between iterations 27 and 29.

Across both datasets the study comprises 2,676 launched chains and approximately 79,000 scored LLM inferences.

3.5 Statistical Analysis

We report bootstrap 95% confidence intervals ($B = 10{,}000$) [Efron and Tibshirani, 1993] on all mean final similarities. Tier differences are tested with Kruskal–Wallis H [Kruskal and Wallis, 1952] and Dunn's post-hoc with Bonferroni correction [Dunn, 1964]. KV cache and quantization-format effects use Wilcoxon signed-rank tests on paired seed-matched deltas. Cross-model prompt rankings use Spearman ρ with bootstrap confidence intervals. Effect sizes are reported as Cohen's d for key contrasts.

Two caveats apply throughout. First, the Kruskal–Wallis and Dunn tests treat individual seed-chains as independent observations, which they are not: chains sharing a model are drawn

from the same generative process, so the nominal p-values are anti-conservative. We report these tests descriptively, to characterize the separation between bands, rather than as inferential claims about tier membership. Second, with $n = 5$ seed-matched pairs per cell the two-sided Wilcoxon test cannot return $p < 0.125$, so per-cell p-values should be read as descriptive rather than as tests.

Because fixing the sampling seed does not pin a chain on these serving stacks (Section 5), the five "seeded" runs behind each condition are treated as five independent samples from the model's output distribution rather than as reproducible replicates. Every bootstrap interval reported here is an interval over those samples.

We formalize *fixed-point attractor* detection as: a run is locked if the range of the last 10 iterations' similarity values is < 0.005.

4 Results

4.1 Cross-Family Architecture Tiers

Figure 1 shows drift trajectories for all 17 models with 95% bootstrap CI bands. A clear four-tier hierarchy emerges:

1. **Stable** (≥ 0.80): Models that maintain high fidelity across 30 iterations (e.g., qwen3:30b-instruct, devstral-small-2:24b).
2. **Moderate** (0.60–0.80): Gradual degradation without catastrophic failure.
3. **Degraded** (0.40–0.60): Substantial meaning loss by iteration 30.
4. **Catastrophic** (< 0.40): Rapid collapse, often at a specific tipping point (e.g., llama3.1:8b loses 0.60 similarity in a single step).

We retain all four band definitions above because they partition the $[0, 1]$ similarity range without gaps, but note that *no model in this study fell in the Degraded band*: the lowest Moderate model sits at 0.609 and the highest Catastrophic model at 0.395. The observed distribution of model means is therefore trimodal, with an empty gap between 0.40 and 0.60, and the Degraded band is retained as a definition rather than as an occupied category. Figure legends mark it accordingly. The gap between the best and worst model means is 0.67—larger than the effect of any other factor measured in this paper, and larger than the entire within-family spread above the stability threshold reported in Section 4.2.

The Kruskal–Wallis test confirms significant tier differences ($H = 57.54$, $p = 3.21 \times 10^{-13}$). Dunn's post-hoc tests with Bonferroni correction show all pairwise tier differences are significant (Stable vs. Catastrophic: $p < 10^{-12}$, $d = 4.70$; Stable vs. Moderate: $p < 10^{-5}$, $d = 1.46$; Moderate vs. Catastrophic: $p < 0.001$), subject to the independence caveat of Section 3.5.

Note that some nominally stable-tier models exhibit high seed variance (e.g., glm-4.7-flash, 95% CI $[0.635, 0.939]$), suggesting that tier membership can be fragile near boundary thresholds; that model is also the one whose chains all terminated early (Table 1), so its mean is taken over chains that stopped between iterations 3 and 9 rather than at iteration 30 and is not comparable to the other rows. Table 1 reports per-model statistics with confidence intervals, the number of chains behind each mean, and how many of those reached iteration 30.

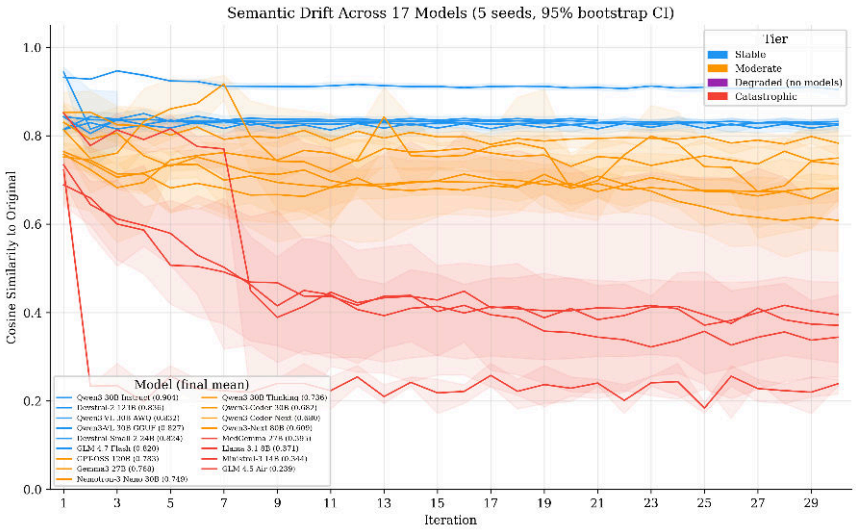

Figure 1: Semantic drift across 17 models. Lines show mean cosine similarity to the original across up to 5 seed-chains; shaded regions indicate 95% bootstrap CI. Llama 3.1 8B and GLM 4.5 Air rest on $n = 3$ (two chains each produced no scored iteration), and the GLM 4.7 Flash curves end early because all five of its chains terminated before iteration 30; see Table 1. The Degraded band (0.40–0.60) is empty in this study and is labelled as such in the legend.

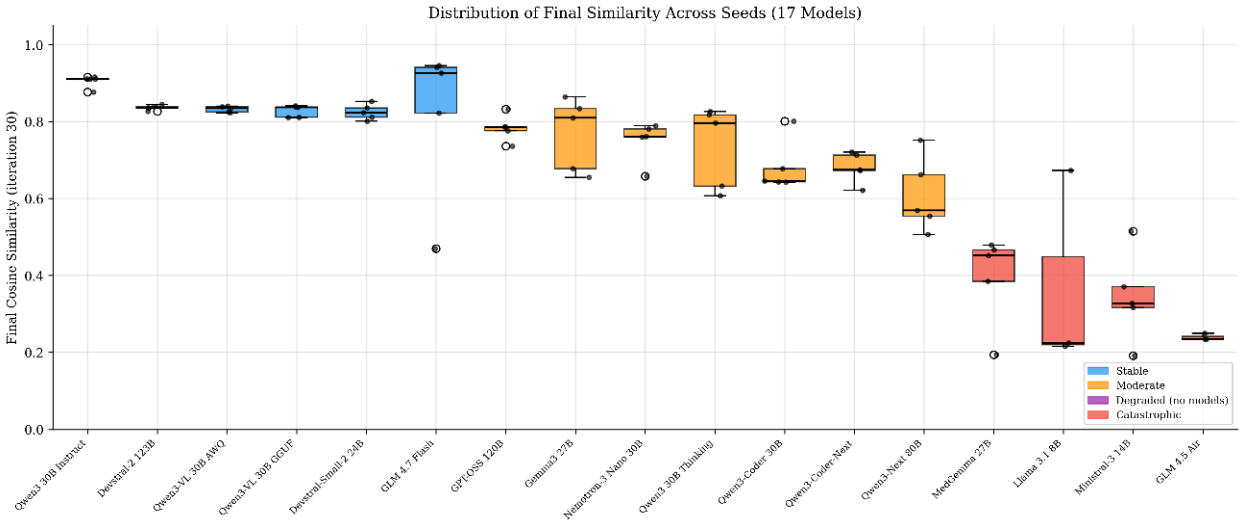

Figure 2: Distribution of final cosine similarity across the available seed-chains for each model (5 unless noted in Table 1; for a chain that terminated early the final value is its last scored iteration rather than iteration 30). Colors indicate stability tier; the Degraded band is empty in this study. Note the high variance in catastrophic-tier models.

Table 1: Per-model drift statistics (RQ1). Mean final cosine similarity with 95% bootstrap CI, sorted by mean descending. Five chains were launched per model; n is the number that returned at least one scored iteration and is the number the mean and CI are computed over, while n_{30} is the number that ran the full 30 iterations. Rows with $n < 5$ or $n_{30} < n$ are marked †: the final similarity for a truncated chain is its last scored iteration, not iteration 30.

Model	Mean	95% CI	n	n_{30}	Tier
Qwen3 30B Instruct	0.904	[0.890, 0.913]	5	5	Stable
Devstral-2 123B	0.836†	[0.831, 0.841]	5	4	Stable
Qwen3-VL 30B AWQ	0.832	[0.826, 0.838]	5	5	Stable
Qwen3-VL 30B GGUF	0.827	[0.816, 0.839]	5	5	Stable
Devstral-Small-2 24B	0.824	[0.809, 0.840]	5	5	Stable
GLM 4.7 Flash	0.820†	[0.635, 0.939]	5	0	Stable
GPT-OSS 120B	0.783	[0.756, 0.812]	5	5	Moderate
Gemma3 27B	0.768†	[0.695, 0.841]	5	4	Moderate
Nemotron-3 Nano 30B	0.749	[0.702, 0.780]	5	5	Moderate
Qwen3 30B Thinking	0.736	[0.654, 0.816]	5	5	Moderate
Qwen3-Coder 30B	0.682	[0.643, 0.744]	5	5	Moderate
Qwen3-Coder-Next	0.680	[0.650, 0.709]	5	5	Moderate
Qwen3-Next 80B	0.609	[0.538, 0.694]	5	5	Moderate
MedGemma 27B	0.395	[0.289, 0.469]	5	5	Catastrophic
Llama 3.1 8B	0.371†	[0.216, 0.673]	3	3	Catastrophic
Ministral-3 14B	0.344	[0.255, 0.440]	5	5	Catastrophic
GLM 4.5 Air	0.239†	[0.234, 0.249]	3	3	Catastrophic

4.2 Within-Family Scaling and Quantization Level

Holding the architecture fixed removes the family confound that limits Section 4.1. Figure 3 shows drift trajectories for all 18 Qwen3.5 GGUF configurations on the spaghetti recipe, and Table 2 gives summary statistics for both recipes with the n behind each cell.

Stability threshold at 4B. Table 2 reports two statistics per cell: the mean over *clean* chains, those that produced text at every iteration, and the rate at which chains instead entered the runaway-thinking failure mode of Section 4.3, after which every remaining iteration is empty and similarity sits at the embedding floor (0.235 for spaghetti, 0.064 for lasagna). Among clean chains, every configuration at or above 4B parameters reaches a mean final similarity between 0.82 and 0.93 on spaghetti, and between 0.74 and 0.86 on lasagna. Below 4B the picture changes in kind rather than degree. The 0.8B models produce no clean lasagna chain at all and only one clean spaghetti chain in ten; the 2B models lose 20–80% of their chains to runaway depending on precision and recipe. A model below the threshold is therefore not merely a noisier paraphraser but one whose chains fail outright in a large fraction of runs.

Non-monotonic precision. Within the clean chains, precision level does not order drift. For the 4B dense model BF16 (0.885) leads Q4_K_M (0.861) and Q8_0 (0.824). For the 27B dense model the order is Q4_K_M (0.914, $n = 3$), BF16 (0.900), Q8_0 (0.871), and for the 35B MoE it is Q4_K_M (0.923, $n = 2$), BF16 (0.846), Q8_0 (0.837). For the 2B model BF16 (0.903, $n = 3$) edges Q8_0 (0.876) on spaghetti, but on lasagna Q8_0 loses four of five chains to runaway while Q4_K_M loses one. Higher precision therefore does not guarantee less drift—the conclusion the KV-cache

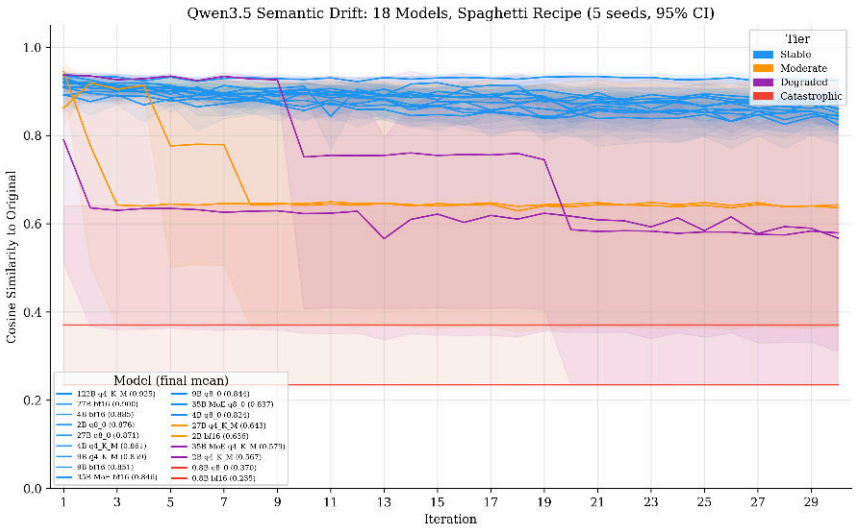

Figure 3: Semantic drift across 18 Qwen3.5 GGUF models (spaghetti recipe, up to 5 completed chains per model, 95% bootstrap CI bands). Models colored by stability tier; per-model n is given in Table 2.

experiment reaches by another route (Section 4.5)—but the comparison must be read together with the failure rates: at 4-bit the clean means are the highest in their size class and rest on the fewest chains, because 40–50% of the 27B and 35B Q4_K_M spaghetti chains ran away. When every completed chain is counted instead, those two cells fall to 0.643 and 0.579 (Table 2, "all" arm in the released statistics), and the apparent precision penalty is entirely a failure-rate penalty.

Mixture-of-experts and the largest model. The 122B MoE achieves the family's best spaghetti score (0.925, CI [0.915, 0.937]) with no runaway chains at any precision, and its lasagna score (0.864) is also the family's best. At matched parameter counts the 35B MoE and the 27B dense model are indistinguishable on clean chains (0.837–0.923 vs. 0.871–0.914 across precisions) and share the same Q4_K_M runaway rate; the sparse-activation advantage suggested by the cross-family tiers is not reproduced within the family.

Statistical tests. Assigning each configuration to a tier by its all-chain spaghetti mean occupies all four tiers (12 Stable, 2 Moderate, 2 Degraded, 2 Catastrophic). Kruskal–Wallis across the occupied tiers gives $H = 18.19$, $p = 4.0 \times 10^{-4}$, with Cohen's $d = 6.18$ between Stable and Catastrophic and 1.67 between Stable and Moderate, subject to the independence caveat of Section 3.5. Because the Moderate and Degraded tiers are populated only by the runaway-affected Q4_K_M and 2B cells, the tier structure within the family is largely a failure-rate structure.

Interaction with stimulus complexity. Figure 4 presents the size × precision heatmap for both recipes, and Figure 5 contrasts them across all 18 configurations. For models above the threshold the lasagna mean is consistently the lower of the two, by 0.03–0.14 (122B: 0.925 vs. 0.864; 27B BF16: 0.900 vs. 0.855; 4B BF16: 0.885 vs. 0.741), the expected cost of carrying the long, nominally 21-step context through 30 rewrites. The runaway failure mode has the opposite dependence: on the 27B and 35B Q4_K_M builds it occurs only on the 3-step spaghetti stimulus and never on lasagna, and on the 35B NVFP4 build (Section 4.4) the spaghetti rate is 80% against 0% on lasagna. Short

Table 2: Qwen3.5 drift by size and quantization. Mean final cosine similarity at iteration 30 over *clean* chains only (chains with no empty, runaway-thinking generation), with 95% bootstrap CI, sorted by clean spaghetti mean descending. n_c is the number of clean 30-iteration chains behind each mean; "RA" is the runaway-thinking rate, i.e. the fraction of completed chains that contained at least one empty generation (runaway chains are excluded from the means). All-chain means are reported in the supplementary `all_stats.json`.

Size	Quant	Spaghetti				Lasagna			Tier
		Mean	95% CI	n_c	RA	Mean	n_c	RA	
122B	q4_K_M	0.925	[0.915, 0.937]	5	0.00 (0/5)	0.864	5	0.00 (0/5)	Stable
35B MoE	q4_K_M	0.923	[0.920, 0.927]	2	0.50 (2/4)	0.812	5	0.00 (0/5)	Stable
27B	q4_K_M	0.914	[0.898, 0.929]	3	0.40 (2/5)	0.829	5	0.00 (0/5)	Stable
0.8B	q8_0	0.912	[0.912, 0.912]	1	0.80 (4/5)	—	0	1.00 (5/5)	Stable
2B	bf16	0.903	[0.892, 0.926]	3	0.40 (2/5)	0.760	2	0.60 (3/5)	Stable
27B	bf16	0.900	[0.886, 0.914]	5	0.00 (0/5)	0.855	5	0.00 (0/5)	Stable
4B	bf16	0.885	[0.873, 0.899]	5	0.00 (0/5)	0.741	5	0.00 (0/5)	Stable
2B	q8_0	0.876	[0.814, 0.931]	5	0.00 (0/5)	0.846	1	0.80 (4/5)	Stable
27B	q8_0	0.871	[0.833, 0.910]	5	0.00 (0/5)	0.839	5	0.00 (0/5)	Stable
4B	q4_K_M	0.861	[0.819, 0.899]	5	0.00 (0/5)	0.823	5	0.00 (0/5)	Stable
9B	q4_K_M	0.859	[0.820, 0.892]	5	0.00 (0/5)	0.830	5	0.00 (0/5)	Stable
9B	bf16	0.851	[0.811, 0.893]	5	0.00 (0/5)	0.815	5	0.00 (0/5)	Stable
35B MoE	bf16	0.846	[0.798, 0.887]	5	0.00 (0/5)	0.797	5	0.00 (0/5)	Stable
9B	q8_0	0.844	[0.783, 0.893]	5	0.00 (0/5)	0.795	5	0.00 (0/5)	Stable
35B MoE	q8_0	0.837	[0.815, 0.858]	5	0.00 (0/5)	0.803	5	0.00 (0/5)	Stable
4B	q8_0	0.824	[0.801, 0.846]	5	0.00 (0/5)	0.788	5	0.00 (0/5)	Stable
2B	q4_K_M	0.789	[0.612, 0.889]	3	0.40 (2/5)	0.772	4	0.20 (1/5)	Moderate
0.8B	bf16	—	—	0	1.00 (5/5)	—	0	1.00 (5/5)	Catastrophic

inputs, not long ones, trigger runaway thinking; the most plausible reading is that a trivial rewriting task gives the model's reasoning process no natural stopping point.

Figure 4: Mean final similarity by model size and quantization level over all completed chains, including runaway chains at the empty-output floor; Table 2 gives the clean-chain means and runaway rates. Left: spaghetti recipe. Right: lasagna recipe. Note that 0.8B Q4_K_M does not exist for Qwen3.5.

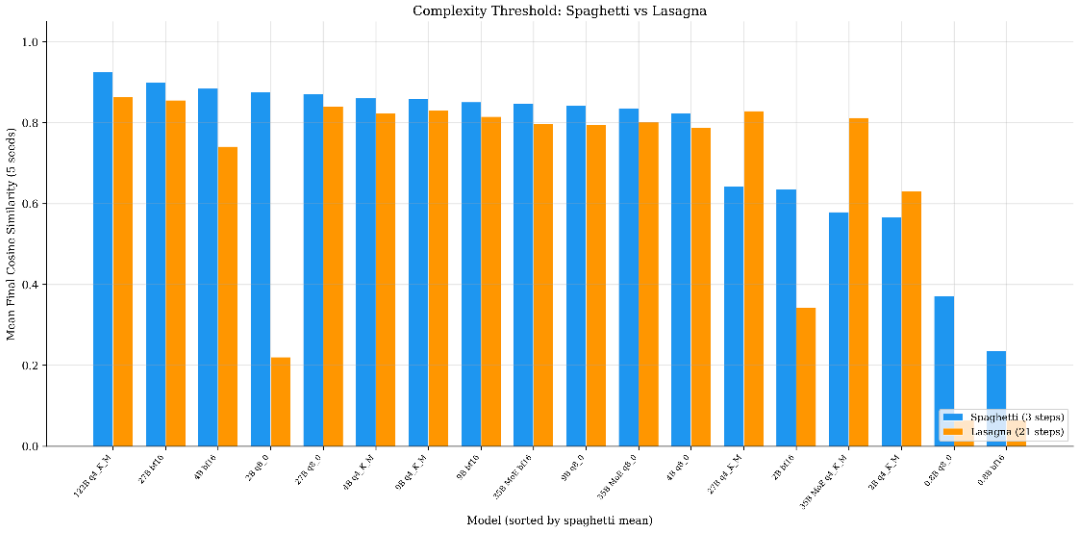

Figure 5: Spaghetti (3 steps) vs. lasagna (21 steps) for all Qwen3.5 GGUF models. Solid bars: spaghetti. Hatched bars: lasagna.

4.3 Runaway Thinking as a Failure Mode

Drift within these chains is not always gradual. A substantial minority of within-family chains fail discretely: the model returns an empty completion, the chain re-feeds the same input at the next iteration, and every subsequent completion is empty as well. Table 3, Table 4 and Figure 6 report the rate at which each condition does so.

What an empty completion is. The empties are not serving errors. Every one is a successful API response whose generation ran for the full 16,384-token budget: on each model the wall time of an empty iteration is constant to within a second and equals the budget divided by that model's measured decode rate (303 s on the 27B GGUF build at 54 tokens/s, 116 s on the 35B MoE GGUF build at 141 tokens/s, 120–260 s on the 35B NVFP4 build). Qwen3.5 enables reasoning by default, and both Ollama and SGLang strip the reasoning block from the returned content. An empty completion is therefore a generation that never left the thinking block: the model reasoned until the budget was exhausted and emitted no answer. We call this *runaway thinking*. The Qwen3.5 embedding appendix (Appendix D) documents the complementary failure at the other extreme, chains whose outputs grow past 6,000 words; that mode is rare in the baseline conditions and is driven by specific system prompts, so this section concentrates on the empty-completion mode.

Rates. Across the 1,969 completed within-family chains, 265 (13.5%, Wilson 95% CI [12.0%, 15.0%]) are runaway. The first-order determinant is whether reasoning was enabled at all. The four vLLM deployments (35B AWQ, 35B GPTQ-Int4, 9B AWQ and the Nemotron NVFP4 control) were production chat servers launched with thinking disabled in the chat template, and across their 993 chains the rate is zero; the Ollama and SGLang deployments ran Qwen3.5 with its default thinking enabled, and their rates are 15–54%. The contrast is therefore between reasoning on and off, not between serving stacks as such: the companion paper [Smith, 2026a] shows that a vLLM deployment of the following generation with thinking switched on does hit the budget, at a lower rate. A second, uncontrolled difference is the sampler. Our requests set only temperature and seed; top-p, top-k and repetition or presence penalties came from each server's defaults, which differ across the three stacks, and the Qwen model cards recommend top-k 20, top-p 0.95 and a presence penalty of 0–2 in thinking mode precisely to prevent endless repetition. The two reasoning-enabled stacks differed in exactly this respect. The Ollama library's Qwen3.5 tags carry a parameter layer that sets top-k 20, top-p 0.95 and a presence penalty of 1.5 (the same layer for every size; our requests overrode only the temperature), so the Ollama arm ran with the vendor's thinking-mode sampler. The SGLang deployment applied no presence penalty. The companion paper tests the sampler directly on the following generation and finds that the presence penalty eliminates the failure there (0 of 15 chains against 8 of 15) while removing top-k and top-p changes nothing. The ordering of the two arms here—15–27% runaway on Ollama against 36–54% on SGLang for the same weights class—is what that result predicts, with the further observation that on Qwen3.5 the penalty mitigates the failure but does not remove it. The rates below should therefore be read as properties of reasoning-enabled Qwen3.5 under a penalised (Ollama) or unpenalised (SGLang) sampler. Within the Ollama sweep the rate then follows size and precision (Table 3): both 0.8B builds fail in 80–100% of chains on either recipe; the 2B builds fail in 20–80%; and above 4B only the Q4_K_M builds fail, at 40% (27B) and 50% (35B MoE) on spaghetti and never on lasagna. The 35B NVFP4 build on SGLang fails in 80% of baseline spaghetti chains and 0% of lasagna chains. Onset is early: the median first empty completion is at iteration 3 (mean 5.7), so a runaway chain typically contributes only a handful of paraphrases before locking.

System prompts trigger it. The system-prompt sweep (Section 4.7) shows that the prompt is as strong a trigger as the model. On the 35B NVFP4 build, ten of the twenty prompts produce runaway in every spaghetti chain and two—Strong Anti-Meta and Ultra-Constrained Fidelity—in every lasagna chain, lifting that configuration's overall rate to 54%. Ultra-Constrained Fidelity and 100% Fidelity raise the 27B GGUF spaghetti rate to 80%, and Ultra-Constrained Fidelity does the same on the 9B GGUF lasagna set, a build that runs away in only 5% of chains otherwise. The prompts that trigger runaway are the ones that most tightly constrain the output format or forbid commentary; an instruction that leaves the model little to say appears to leave its reasoning process with no way to conclude. On the vLLM arms, where reasoning is disabled by the serving configuration, the same prompts produce no empties at all.

Consequence for every other statistic. Because a runaway chain sits at the embedding floor from its onset, a mean over all completed chains mixes a fidelity measurement with a failure rate. Every table in this paper therefore reports the clean-chain mean as the primary statistic with the runaway rate beside it; the all-chain means are retained in the released statistics. The two views can disagree sharply on the same cell—0.923 clean against 0.579 all for the 35B Q4_K_M spaghetti condition—and a reader interested in deployment risk should weigh the rate at least as heavily as the mean.

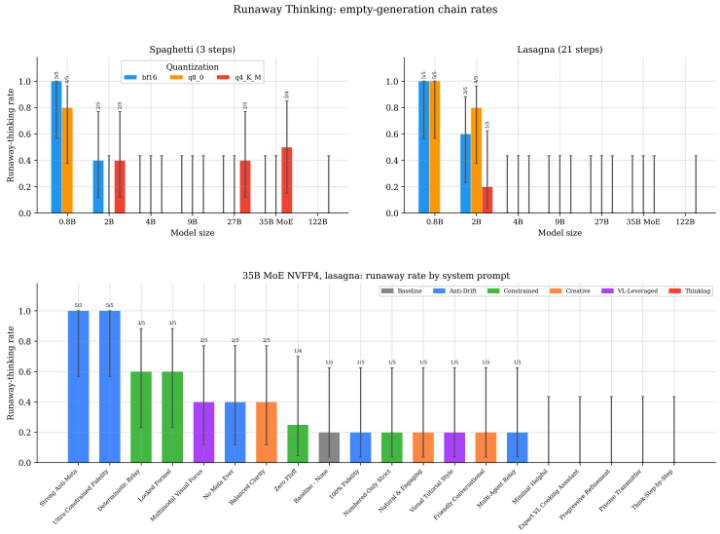

Figure 6: Runaway-generation rates by condition.

4.4 Quantization Format: GGUF vs. AWQ vs. NVFP4 vs. GPTQ-Int4

Section 4.2 varies the precision level within one format family. Here we hold both the weights and the precision fixed and vary only the quantization *algorithm*. Figure 7 compares the 35B MoE model under four 4-bit quantizations of the same underlying Qwen3.5-35B-A3B architecture: GGUF Q4_K_M [Gerganov and llama.cpp contributors, 2023] served by Ollama [Ollama, 2026], AWQ 4-bit [Lin et al., 2024] and GPTQ-Int4 [Frantar et al., 2023] served by vLLM [Kwon et al., 2023], and NVFP4 [NVIDIA et al., 2025] served by SGLang [Zheng et al., 2024]. Because the quantization algorithm and the serving backend covary across three of these four arms, the comparison bounds their combined effect rather than attributing it to either alone; the 9B AWQ arm and the Nemotron NVFP4 control provide two further points at which format and backend can be partially separated.

Table 3: Runaway-thinking rates. A chain is *runaway* when at least one of its 30 iterations returned an empty assistant message after a full-length generation, i.e. the entire token budget was consumed by reasoning; such chains never recover, because the same input is re-fed at the next iteration. Rate = runaway chains / completed chains, with a Wilson 95% interval. Top block: RQ1 size × quantization set; per-prompt rates are in Table 4.

Size	Quant	Spaghetti		Lasagna	
		Rate	95% CI	Rate	95% CI
0.8B	q8_0	0.80 (4/5)	[0.38, 0.96]	1.00 (5/5)	[0.57, 1.00]
0.8B	bf16	1.00 (5/5)	[0.57, 1.00]	1.00 (5/5)	[0.57, 1.00]
2B	q4_K_M	0.40 (2/5)	[0.12, 0.77]	0.20 (1/5)	[0.04, 0.62]
2B	q8_0	0.00 (0/5)	[0.00, 0.43]	0.80 (4/5)	[0.38, 0.96]
2B	bf16	0.40 (2/5)	[0.12, 0.77]	0.60 (3/5)	[0.23, 0.88]
4B	q4_K_M	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
4B	q8_0	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
4B	bf16	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
9B	q4_K_M	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
9B	q8_0	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
9B	bf16	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
27B	q4_K_M	0.40 (2/5)	[0.12, 0.77]	0.00 (0/5)	[0.00, 0.43]
27B	q8_0	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
27B	bf16	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
35B MoE	q4_K_M	0.50 (2/4)	[0.15, 0.85]	0.00 (0/5)	[0.00, 0.43]
35B MoE	q8_0	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
35B MoE	bf16	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]
122B	q4_K_M	0.00 (0/5)	[0.00, 0.43]	0.00 (0/5)	[0.00, 0.43]

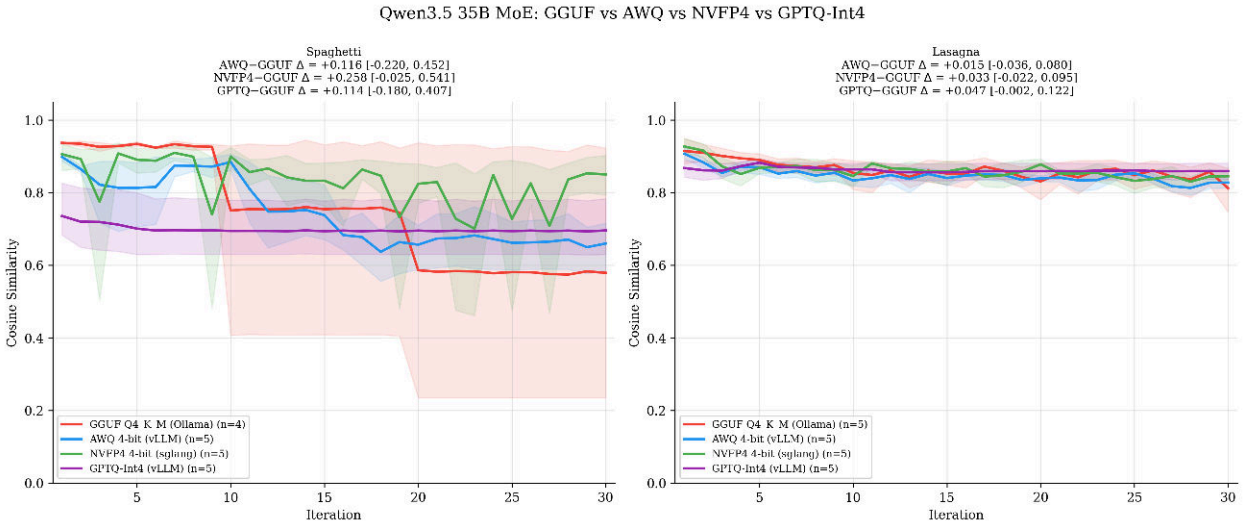

Figure 7: Four quantization methods for the 35B MoE model: GGUF Q4_K_M (Ollama), AWQ 4-bit (vLLM), NVFP4 4-bit (SGLang), and GPTQ-Int4 (vLLM). Left: spaghetti. Right: lasagna. Panel titles give the seed-matched paired delta against GGUF with its bootstrap 95% CI; n per arm is shown in the legend.

Table 2 and Figure 7 give the result. On the lasagna recipe the four formats are close: GGUF Q4_K_M 0.812, AWQ 0.828, NVFP4 0.845 and GPTQ-Int4 0.860, with seed-matched paired deltas against GGUF of +0.015, +0.033 and +0.047 whose bootstrap intervals all include zero, and no runaway chains in any arm. On the spaghetti recipe the formats separate. Among clean chains GGUF Q4_K_M is best (0.923) and the two vLLM formats are far below it (AWQ 0.660, CI [0.586, 0.716]; GPTQ-Int4 0.696, CI [0.630, 0.784]), a gap of 0.23–0.26 that is the largest within-family effect in this paper apart from the capacity threshold. NVFP4 falls between (0.757), but on a single clean chain. The GGUF and NVFP4 clean means rest on two and one chain respectively because those two arms lose 50% and 80% of their spaghetti chains to runaway thinking (Section 4.3), while the vLLM arms lose none; the paired deltas computed over all completed chains (+0.116 AWQ, +0.258 NVFP4, +0.114 GPTQ, all intervals spanning zero at $n = 4$ pairs) accordingly point the other way. The format comparison on the short stimulus is thus a trade: the GGUF build paraphrases most faithfully when it answers, and fails to answer half the time.

Two further arms help separate format from serving stack. The 9B AWQ build on vLLM behaves like the 35B AWQ build (spaghetti 0.683, lasagna 0.877 at $T = 0.7$) rather than like the 9B GGUF build (0.859, 0.830), so the spaghetti penalty follows the format and stack, not the size. The cross-family dataset shows the same divergence in a second family: the AWQ and GGUF builds of Qwen3-VL 30B, both at 4-bit, produce different drift dynamics (Section 4.1). Because the quantization algorithm and the serving backend covary in three of the four arms, we attribute the effect to the algorithm-and-stack combination rather than to either alone. The practical conclusion does not depend on the attribution: 4-bit builds of the same weights are not interchangeable in a drift-sensitive pipeline, and the build that will be deployed is the one that must be benchmarked.

4.5 KV Cache Precision

Figure 8 shows the paired delta (fp16 − q8_0 KV cache) for 10 Qwen3 models with bootstrap CIs and Wilcoxon p-values.

Key findings:

- The overall effect is not statistically significant (Wilcoxon $p = 0.375$, $n = 10$), reflecting directionally mixed, model-dependent responses that cancel in aggregate.
- The per-model effects are directionally mixed rather than demonstrably non-monotonic: eight of the ten per-model delta confidence intervals include zero. Only 8B q4_K_M (+0.128, CI $[0.050, 0.214]$) and 4B Instruct q4_K_M (+0.011, CI $[0.001, 0.022]$) have intervals excluding zero, and the latter is negligible in magnitude. With $n = 5$ seed-matched pairs per model, the per-model p-values in Table 5 should not be read as tests (Section 3.5).
- Small models (0.6B) show the largest point improvement from fp16 KV (+0.257 and +0.131), consistent with KV cache noise disproportionately affecting models with fewer parameters, though both intervals span zero.
- The 80B model shows the largest point *regression* with fp16 KV (−0.100, CI $[-0.259, 0.079]$), which would be consistent with a latent attractor that q8 KV noise was masking; on this evidence the direction is suggestive rather than established.
- Among the q4_K_M models the deltas are small in magnitude, consistent with weight quantization noise dominating KV cache noise.

Figure 9 shows the heatmap of deltas across model size and weight quantization. The magnitudes involved are an order below the cross-family spread of Section 4.1 and comparable to the temperature effects of Section 4.6.

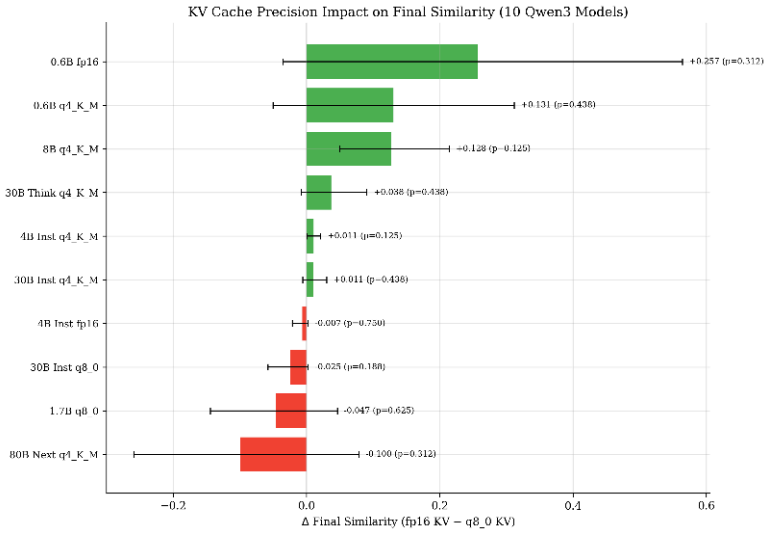

Figure 8: KV cache precision impact (fp16 − q8_0) on final similarity. Error bars show 95% bootstrap CI. Green = improvement, red = regression. Annotated with Wilcoxon p-values; with five pairs the smallest attainable two-sided p is 0.125, so the annotations are descriptive.

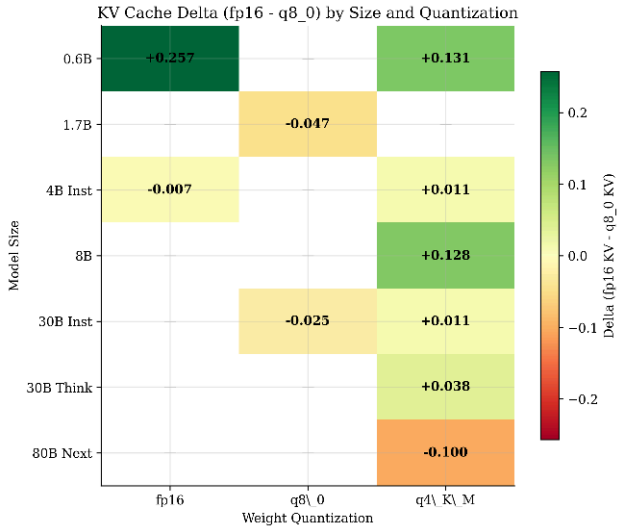

Figure 9: KV cache precision delta across model size and weight quantization. Green = fp16 KV helped; red = hurt. Blank cells indicate untested combinations.

4.6 Temperature

Temperature was ablated in both datasets. Figure 10 shows the cross-family interaction between temperature and model, and Figure 11 the within-family version across eight Qwen3.5 configurations

(4 Ollama GGUF, the 35B AWQ, NVFP4 and GPTQ-Int4 variants, the 9B AWQ variant, and the Nemotron control).

Cross-family observations.

- $T = 0$ (greedy decoding) does not guarantee stability: catastrophic models still drift, confirming that drift is not merely a function of sampling noise.
- Stable models (qwen3:30b-instruct) are robust across all temperatures.
- Unstable models (llama3.1:8b) are sensitive to temperature, with higher temperatures accelerating collapse.
- The relationship is non-linear: moderate temperatures ($T = 0.3$–0.5) sometimes *improve* stability relative to $T = 0$, suggesting that some sampling diversity can help escape local attractors.

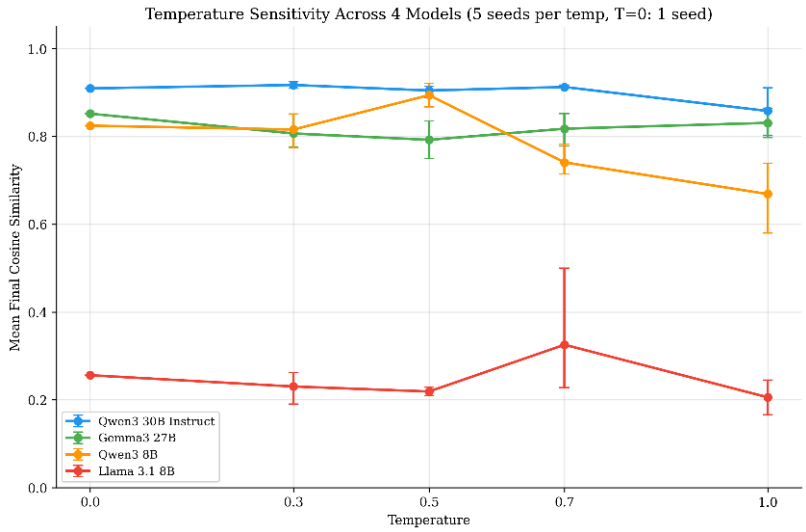

Figure 10: Temperature sensitivity across 4 cross-family models. Error bars show 95% CI (5 seeds; $T = 0$ is deterministic, single run). Drift is not reducible to sampling entropy.

Size-dependent sensitivity within the family. The 9B GGUF model shows remarkable temperature insensitivity: final similarity ranges only 0.823–0.890 across $T \in [0.0, 1.0]$ on spaghetti. The 2B and 35B MoE GGUF models appear at first to show *inverted* sensitivity: their all-chain means sit at the empty-output floor of 0.235 at $T \leq 0.3$ (2B) and $T \leq 0.5$ (35B) and rise monotonically with temperature (2B: 0.512, 0.567, 0.691 at $T = 0.5$, 0.7, 1.0; 35B: 0.899 at $T = 1.0$). Reading those cells through the runaway classification of Section 4.3 shows that this is not a drift effect. At low temperature every chain of both models ran away—the runaway rate is 100% at $T \leq 0.3$ for the 2B model and at $T \leq 0.5$ for the 35B MoE—and the rate falls as temperature rises (2B: 60%, 40%, 0% at $T = 0.5$, 0.7, 1.0; 35B: 60% at $T = 0.7$, 0% at $T = 1.0$). Among the chains that do not run away, similarity is high and roughly flat (2B 0.93 at $T = 0.5$; 35B 0.92 at $T = 0.7$). Low temperature, in other words, makes reasoning-enabled small and 4-bit models more likely to loop inside the thinking block until the budget is exhausted, which is the behaviour the Qwen model cards warn greedy decoding produces in thinking mode; it does not make their paraphrases worse. Below the capacity threshold, temperature acts on the failure rate rather than on fidelity.

Table 4: Runaway-thinking rate per system prompt (up to 5 chains per cell; rate = runaway chains / completed chains).

System prompt	35B MoE NVFP4		27B GGUF Q4_K_M	
Prompt	Spag.	Las.	Spag.	Las.
Baseline - None	0.80 (4/5)	0.20 (1/5)	0.60 (3/5)	0.00 (0/5)
Minimal Helpful	0.60 (3/5)	0.00 (0/5)	0.20 (1/5)	0.00 (0/5)
Strong Anti-Meta	1.00 (5/5)	1.00 (5/5)	0.40 (2/5)	0.40 (2/5)
Ultra-Constrained Fidelity	1.00 (5/5)	1.00 (5/5)	0.80 (4/5)	0.60 (3/5)
Expert VL Cooking Assistant	0.60 (3/5)	0.00 (0/5)	0.20 (1/5)	0.20 (1/5)
Deterministic Relay	0.00 (0/5)	0.60 (3/5)	0.40 (2/5)	0.40 (2/5)
100% Fidelity	1.00 (5/5)	0.20 (1/5)	0.80 (4/5)	0.60 (3/5)
Numbered-Only Strict	1.00 (5/5)	0.20 (1/5)	0.80 (4/5)	0.00 (0/5)
Natural & Engaging	1.00 (5/5)	0.20 (1/5)	0.40 (2/5)	0.20 (1/5)
Progressive Refinement	1.00 (5/5)	0.00 (0/5)	0.40 (2/5)	0.00 (0/5)
Visual Tutorial Style	0.40 (2/5)	0.20 (1/5)	0.00 (0/5)	0.20 (1/5)
Precise Transmitter	0.60 (3/5)	0.00 (0/5)	0.00 (0/5)	0.00 (0/5)
Zero Fluff	1.00 (5/5)	0.25 (1/4)	0.20 (1/5)	0.40 (2/5)
Locked Format	1.00 (5/5)	0.60 (3/5)	0.60 (3/5)	0.20 (1/5)
Friendly Conversational	0.20 (1/5)	0.20 (1/5)	0.20 (1/5)	0.00 (0/1)
Multi-Agent Relay	0.80 (4/5)	0.20 (1/5)	0.00 (0/5)	0.00 (0/3)
Think-Step-by-Step	1.00 (5/5)	0.00 (0/5)	0.40 (2/5)	0.00 (0/2)
Multimodal Visual Focus	0.80 (4/5)	0.40 (2/5)	0.00 (0/5)	0.00 (0/3)
No Meta Ever	0.80 (4/5)	0.40 (2/5)	0.60 (3/5)	0.00 (0/2)
Balanced Clarity	1.00 (5/5)	0.40 (2/5)	0.20 (1/5)	—

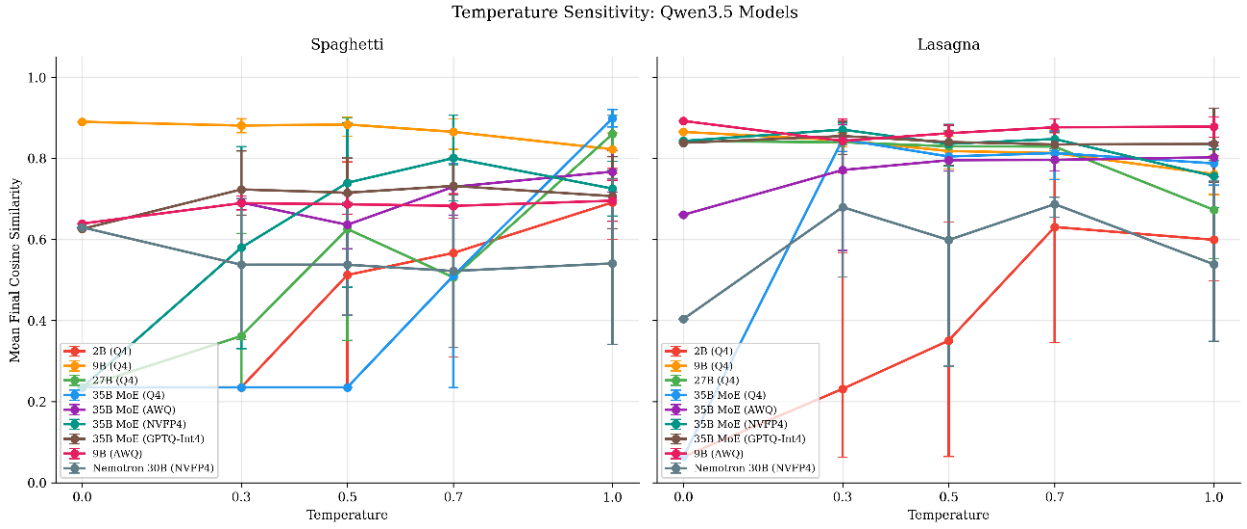

Figure 11: Temperature sensitivity across Qwen3.5 configurations. Left: spaghetti. Right: lasagna.

Table 5: KV cache precision effects (RQ2). Paired delta = fp16 KV − q8_0 KV on final cosine similarity, 5 seed-matched pairs per model, with the bootstrap 95% CI of the delta ($B = 10{,}000$). With $n = 5$ pairs the two-sided Wilcoxon signed-rank test cannot return a p below 0.125, so the p column is reported for completeness only and no per-model test here is capable of reaching $p < 0.05$; the delta CIs are the informative quantity.

Model	Size	Quant	q8 Mean	fp16 Mean	Δ	95% CI of Δ	p
0.6B fp16	0.6B	fp16	0.577	0.834	+0.257	[-0.035, 0.564]	0.312
0.6B q4_K_M	0.6B	q4_K_M	0.674	0.805	+0.131	[-0.050, 0.312]	0.438
8B q4_K_M	8B	q4_K_M	0.731	0.858	+0.128	[0.050, 0.214]	0.125
30B Think q4_K_M	30B Think	q4_K_M	0.742	0.779	+0.038	[-0.008, 0.091]	0.438
4B Inst q4_K_M	4B Inst	q4_K_M	0.918	0.929	+0.011	[0.001, 0.022]	0.125
30B Inst q4_K_M	30B Inst	q4_K_M	0.907	0.917	+0.011	[-0.005, 0.031]	0.438
4B Inst fp16	4B Inst	fp16	0.897	0.890	-0.007	[-0.021, 0.003]	0.750
30B Inst q8_0	30B Inst	q8_0	0.912	0.887	-0.025	[-0.057, 0.003]	0.188
1.7B q8_0	1.7B	q8_0	0.767	0.720	-0.047	[-0.144, 0.046]	0.625
80B Next q4_K_M	80B Next	q4_K_M	0.609	0.508	-0.100	[-0.259, 0.079]	0.312

Format-dependent temperature profiles. The 35B AWQ model shows a qualitatively different pattern: similarity *increases* with temperature on both recipes (spaghetti: 0.636 at $T = 0.5$ to 0.767 at $T = 1.0$; lasagna: 0.661 at $T = 0.0$ to 0.803 at $T = 1.0$; the AWQ $T = 0.0$ spaghetti chain was among those lost to the failed rerun). The 35B GPTQ-Int4 and 9B AWQ variants are by contrast nearly flat across the whole temperature range (GPTQ spaghetti 0.626–0.732; 9B AWQ spaghetti 0.639–0.696). This is the opposite of the conventional expectation that higher temperature increases randomness and thus drift. We hypothesize that AWQ quantization creates a restricted output distribution at low temperatures, and higher temperature helps the model access more of its learned paraphrase space.

4.7 System Prompts: Complexity Threshold and Rank Inversion

The same 20 system prompts (Appendix A) were run in both datasets: across families on 30B AWQ and 8B q4_K_M, and within the Qwen3.5 family on six configurations—9B and 27B GGUF, 35B AWQ, 35B NVFP4, 35B GPTQ-Int4 and 9B AWQ—plus the out-of-family Nemotron-3-Nano-30B control. Each condition uses up to 5 completed chains on both recipes. Table 6 summarizes the per-configuration spread, best and worst prompt, and the pairwise Spearman rank correlations, and Table 7 gives the per-prompt means.

Complexity threshold. Figure 12 shows that on the 3-step spaghetti recipe, all 20 system prompts produce nearly identical drift (spread < 0.01 on 30B AWQ). On the long lasagna recipe (nominally 21 steps), the spread increases to $\sim$0.054, revealing a *complexity threshold* below which prompt engineering is invisible.

Prompt effectiveness. Figure 13 shows drift trajectories for all 20 system prompts on the 30B AWQ model with lasagna, and Figure 14 the corresponding within-family view on Qwen3.5 27B GGUF. The best-performing prompts on 30B AWQ (Multi-Agent Relay, No Meta Ever) target a specific failure mode: meta-commentary about the relay task itself. Over-constraining (Ultra-Constrained Fidelity) backfires, producing the worst performance.

Cross-model rank inversion. Figure 15 reveals that prompt rankings on 30B AWQ are essentially uncorrelated with rankings on 8B q4_K_M (Spearman $\rho = 0.15$, $p = 0.53$, 95% CI $[-0.37, 0.61]$), so prompt optimization does not transfer across families. Figure 16 extends the same analysis within one family and across quantization formats of one checkpoint.

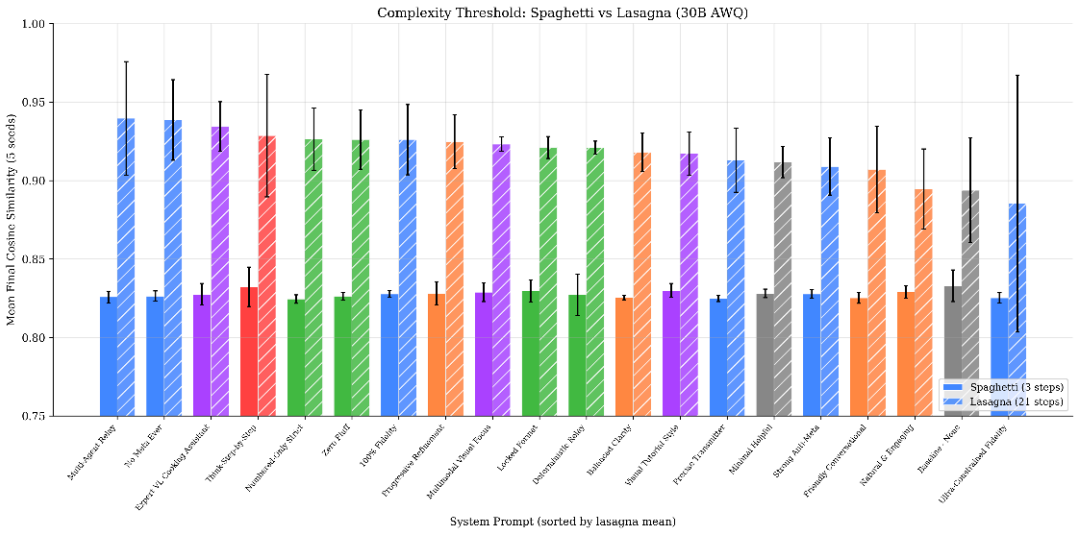

Figure 12: Complexity threshold: system prompts are invisible on the 3-step recipe (solid bars) but differentiate on the long recipe, nominally 21 steps (hatched bars). 30B AWQ, 5 seeds.

The complexity threshold is a property of the model, not of the task. On the cross-family 30B AWQ model the 20 prompts are indistinguishable on the 3-step recipe (spread 0.008) and separate only on the long recipe (spread 0.054), the pattern Figure 12 shows. On the cross-family

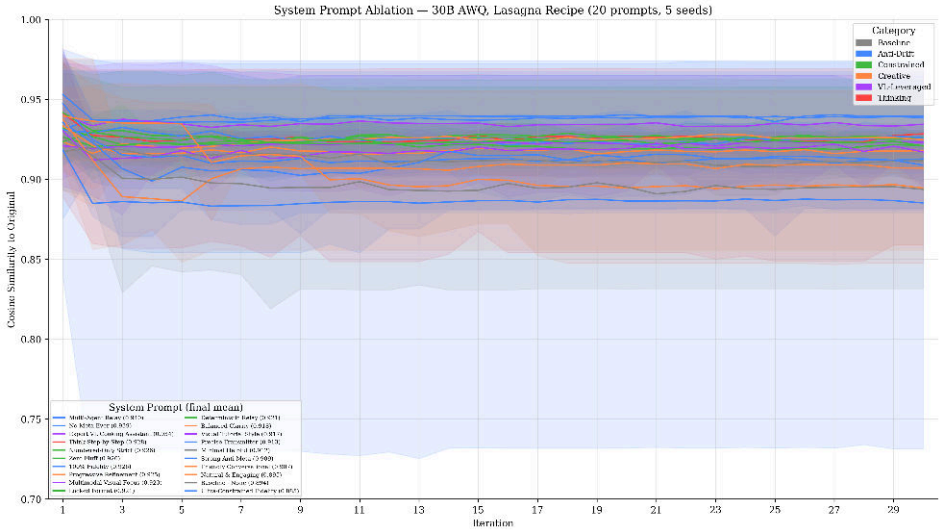

Figure 13: System prompt ablation on 30B AWQ, lasagna recipe. Lines colored by category; shaded = min/max across 5 seeds.

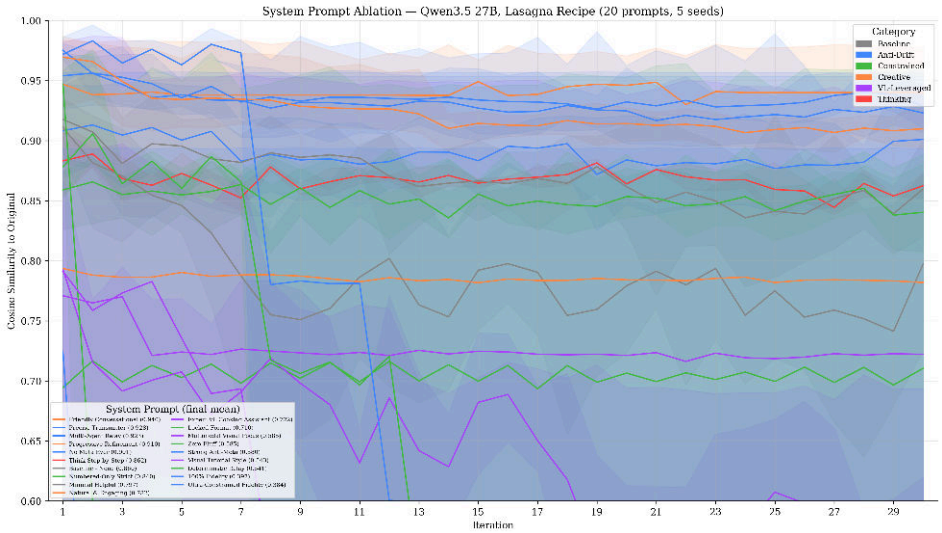

Figure 14: System prompt ablation on Qwen3.5 27B GGUF, lasagna recipe. 20 prompts shown with mean ± min/max band over the completed chains for each prompt. Prompts 15–20 rest on fewer than 5 chains and "Balanced Clarity" on none (Section 3.4).

8B model the spreads are 0.154 and 0.098, and within the Qwen3.5 family they are larger still and no longer ordered by recipe: 0.30–0.49 on spaghetti and 0.29–0.74 on lasagna across the six configurations (Table 6), with the Nemotron control at 0.50 and 0.56. The threshold therefore describes a model that is already near its ceiling, where no prompt can help and only a long input gives one room to hurt. For a configuration that drifts on its own, prompts matter on any input.

Which prompts help, and which break the model. Where prompts matter, the fidelity-oriented prompts win: 100% Fidelity, Precise Transmitter and Ultra-Constrained Fidelity take the best position on nine of the twelve Qwen3.5 configuration–recipe pairs, reaching 0.96–0.99, and the

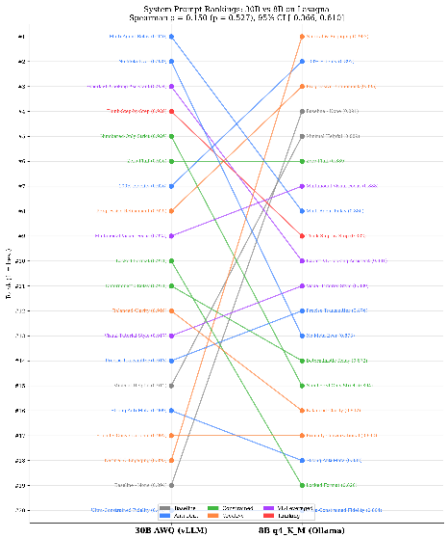

Figure 15: System prompt rankings are uncorrelated across model sizes. Slopes show rank changes between 30B (left) and 8B (right) on lasagna.

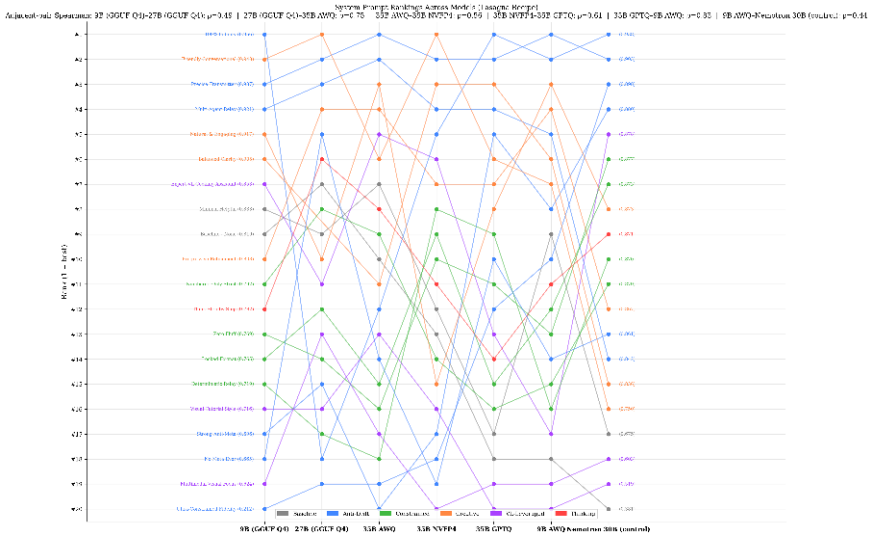

Figure 16: System prompt rankings across all six Qwen3.5 configurations and the Nemotron control (lasagna). Adjacent-column Spearman ρ values are annotated; the full matrix with bootstrap CIs is in Table 6.

visually oriented prompts (Visual Tutorial Style, Multimodal Visual Focus) take the worst position on nine of the twelve, at 0.59–0.67, below the no-prompt baseline. The cross-family 30B AWQ model is the exception in both directions: its best prompts are the anti-meta-commentary ones (Multi-Agent Relay, No Meta Ever) and Ultra-Constrained Fidelity is its worst. That prompt is also the one that most reliably triggers runaway thinking on the reasoning-enabled builds (Section 4.3): it is the best prompt on the 35B GPTQ-Int4 and 9B AWQ builds served by vLLM and a guaranteed failure on the 35B NVFP4 build served by SGLang. The same words are the best or the worst instruction depending on the stack they run on.

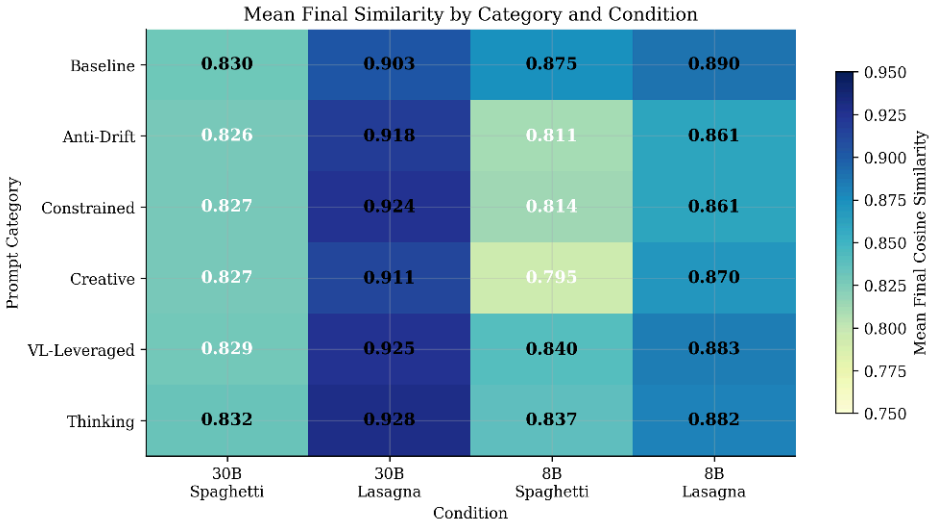

Figure 17: Mean final similarity by prompt category and cross-family condition.

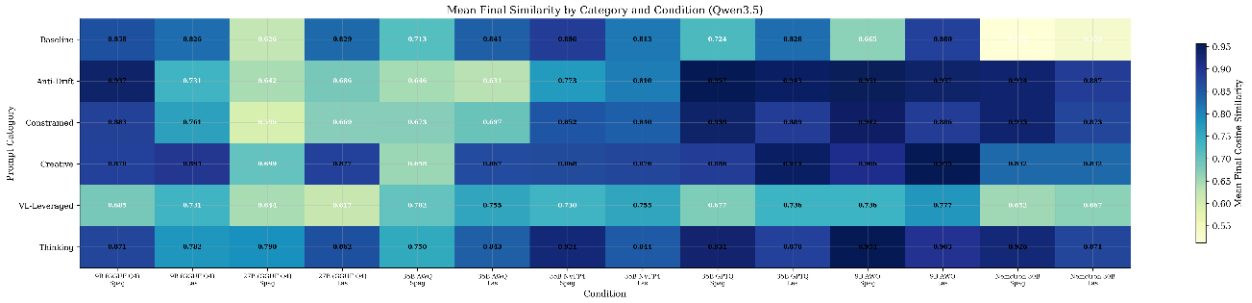

Figure 18: Mean final similarity by prompt category and Qwen3.5 condition.

Rankings transfer between related configurations only. Across families, prompt rankings on 30B AWQ and 8B q4_K_M are uncorrelated (Spearman $\rho = 0.15$, $p = 0.53$, 95% CI $[-0.37, 0.61]$; Figure 15). Within the Qwen3.5 family the picture is graded rather than binary (Table 6, Figure 16). The strongest agreement is between the two vLLM builds of different sizes, 35B GPTQ-Int4 and 9B AWQ ($\rho = 0.83$, CI [0.54, 0.94]), and between the 9B GGUF build and the two 35B non-GGUF builds (0.78 and 0.80). Agreement between formats of the same 35B checkpoint is weaker (AWQ–GPTQ 0.26, AWQ–NVFP4 0.56, NVFP4–GPTQ 0.61), and between the two GGUF sizes weaker still (9B–27B 0.49). Every correlation with the out-of-family Nemotron control is weak (-0.33 to 0.50) and every one of those intervals includes zero. Prompt rankings, then, are neither universal nor arbitrary: they are shared by configurations that share a serving stack and a family, and they do not survive a change of family. Table 7 gives the per-prompt means and Figures 17 and 18 the category view, in which the VL-Leveraged category is the worst on every one of the fourteen within-family columns.

Table 6: RQ4 summary (lasagna recipe), computed over *clean* chains only. Spread is the difference between the best- and worst-performing system prompt for that configuration; $n_{\min}$ is the smallest number of clean chains behind any prompt cell; "RA" is the configuration-wide runaway-thinking rate over all 20 prompts × 5 seeds.

Configuration	Mean	Spread	Best prompt	Worst prompt	$n_{\min}$	RA
9B (GGUF Q4)	0.822	0.332	100% Fidelity (0.956)	Multimodal Visual Focus (0.624)	1	0.05 (5/97)
27B (GGUF Q4)	0.868	0.380	Natural & Engaging (0.965)	Multimodal Visual Focus (0.585)	1	0.20 (16/81)
35B AWQ	0.742	0.743	Precise Transmitter (0.917)	Strong Anti-Meta (0.175)	5	0.00 (0/100)
35B NVFP4	0.854	0.294	Balanced Clarity (0.956)	Multimodal Visual Focus (0.663)	2	0.30 (30/99)
35B GPTQ	0.887	0.348	100% Fidelity (0.981)	Visual Tutorial Style (0.633)	4	0.00 (0/99)
9B AWQ	0.899	0.300	Precise Transmitter (0.972)	Visual Tutorial Style (0.672)	4	0.00 (0/99)
Nemotron 30B (control)	0.803	0.556	100% Fidelity (0.938)	Baseline - None (0.381)	5	0.00 (0/100)

Pairwise Spearman ρ between per-prompt mean rankings (20 prompts, lasagna), with 95% bootstrap CIs ($B = 10{,}000$).

Configuration A	Configuration B	ρ	95% CI
9B (GGUF Q4)	27B (GGUF Q4)	0.486	[-0.075, 0.897]
9B (GGUF Q4)	35B AWQ	0.779	[0.461, 0.918]
9B (GGUF Q4)	35B NVFP4	0.798	[0.444, 0.944]
9B (GGUF Q4)	35B GPTQ	0.549	[0.070, 0.825]
9B (GGUF Q4)	9B AWQ	0.633	[0.209, 0.857]
9B (GGUF Q4)	Nemotron 30B (control)	0.081	[-0.488, 0.573]
27B (GGUF Q4)	35B AWQ	0.746	[0.420, 0.897]
27B (GGUF Q4)	35B NVFP4	0.409	[-0.171, 0.819]
27B (GGUF Q4)	35B GPTQ	0.209	[-0.326, 0.707]
27B (GGUF Q4)	9B AWQ	0.312	[-0.202, 0.714]
27B (GGUF Q4)	Nemotron 30B (control)	-0.333	[-0.781, 0.204]
35B AWQ	35B NVFP4	0.556	[0.123, 0.800]
35B AWQ	35B GPTQ	0.259	[-0.200, 0.644]
35B AWQ	9B AWQ	0.480	[0.012, 0.794]
35B AWQ	Nemotron 30B (control)	-0.093	[-0.560, 0.390]
35B NVFP4	35B GPTQ	0.611	[0.180, 0.855]
35B NVFP4	9B AWQ	0.523	[0.080, 0.785]
35B NVFP4	Nemotron 30B (control)	0.162	[-0.369, 0.618]
35B GPTQ	9B AWQ	0.829	[0.542, 0.937]
35B GPTQ	Nemotron 30B (control)	0.496	[-0.060, 0.839]
9B AWQ	Nemotron 30B (control)	0.441	[-0.071, 0.784]

Table 7: System prompt effects on the lasagna recipe across all tested Qwen3.5 configurations plus the out-of-family Nemotron control. Mean final cosine similarity over the *clean* chains for that cell (chains containing no empty, runaway-thinking generation; up to 5 per cell). A superscript gives n_c when fewer than 5 clean chains survive; "—" marks a cell with no clean chain at all. The last two rows give the smallest per-cell n_c and the configuration-wide runaway-thinking rate. Sorted by the mean across configurations.

Prompt	Category	9B (GGUF Q4)	27B (GGUF Q4)	35B AWQ	35B NVFP4	35B GPTQ	9B AWQ	Nemotron 30B	Mean
Precise Transmitter	Anti-Drift	0.937	0.928	0.917	0.941	0.976	0.972^{4}	0.903	0.939
Natural & Engaging	Creative	0.917	0.965^{4}	0.914	0.929^{4}	0.932	0.968	0.873	0.928
100% Fidelity	Anti-Drift	0.956^{4}	0.917^{2}	0.806	0.881^{4}	0.981	0.970	0.938	0.921
Multi-Agent Relay	Anti-Drift	0.921	0.923^{3}	0.915	0.917^{4}	0.959	0.964	0.842	0.920
Balanced Clarity	Creative	0.905^{3}	—	0.810	0.956^{3}	0.969	0.959	0.836	0.906
Friendly Conversational	Creative	0.948	0.940^{1}	0.869	0.953^{4}	0.940	0.924	0.750	0.903
Progressive Refinement	Creative	0.808	0.910	0.876	0.863	0.936	0.967	0.867	0.890
Expert VL Cooking Assistant	VL-Leveraged	0.853	0.890^{4}	0.872	0.869	0.881	0.865	0.878	0.873
Think-Step-by-Step	Thinking	0.782	0.862^{2}	0.843	0.844	0.878	0.903	0.871	0.855
Numbered-Only Strict	Constrained	0.792	0.840	0.842	0.811^{4}	0.872	0.881	0.870	0.844
Zero Fluff	Constrained	0.769	0.941^{3}	0.660	0.854^{3}	0.903	0.893	0.877	0.842
Locked Format	Constrained	0.765	0.875^{4}	0.745	0.846^{2}	0.874	0.893	0.873	0.839
No Meta Ever	Anti-Drift	0.665	0.901^{2}	0.746	0.841^{3}	0.907	0.884	0.861	0.829
Minimal Helpful	Baseline	0.833	0.797	0.859	0.814	0.832	0.909	0.675	0.817
Deterministic Relay	Constrained	0.719	0.868^{3}	0.542	0.861^{2}	0.907	0.875	0.870	0.806
Strong Anti-Meta	Anti-Drift	0.857^{4}	0.934^{3}	0.175	—	0.943	0.924	0.880	0.786
Ultra-Constrained Fidelity	Anti-Drift	0.858^{1}	0.886^{2}	0.228	—	0.892	0.907	0.898	0.778
Baseline - None	Baseline	0.819	0.860	0.822	0.813^{4}	0.823	0.851	0.381	0.767
Visual Tutorial Style	VL-Leveraged	0.716	0.666^{4}	0.766	0.710^{4}	0.633	0.672	0.519	0.669
Multimodal Visual Focus	VL-Leveraged	0.624	0.585^{3}	0.626	0.663^{3}	0.694^{4}	0.795	0.603	0.656
Minimum n_c		1	0	5	0	4	4	5	
Runaway rate		0.05	0.20	0.00	0.30	0.00	0.00	0.00	

4.8 Embedding Model Validation

To validate that our results are not an artifact of the embedding model, we re-embedded all saved texts with bge-m3 [Chen et al., 2024] in addition to the primary Qwen3-Embedding-0.6B [Zhang et al., 2025], separately for each dataset.

On the cross-family data (Figure 19), the Pearson correlation is $r = 0.960$ and Spearman $\rho = 0.857$ ($n = 21{,}636$ pairs, both $p < 10^{-10}$).

On the within-family data, 20,072 of 22,167 pairs (90.6%) produced valid similarities from both models, yielding $r = 0.825$ and $\rho = 0.815$ ($p < 10^{-10}$ for both). The remaining 2,095 pairs (9.4%) failed because bge-m3's serving endpoint rejects inputs exceeding its 8,192-token context window rather than truncating them, whereas Qwen3-Embedding-0.6B accepts 32,768 tokens. All failed pairs are outputs exceeding 6,000 words, produced by the runaway text expansion described in Section 4.3; their mean Qwen similarity is 0.515 (median 0.572) against a dataset mean of 0.829, confirming that they are systematically the most-drifted outputs. Because both embedding models would score these highly-diverged texts as low similarity, their exclusion is unlikely to meaningfully affect the correlation. Details are in Appendix D.

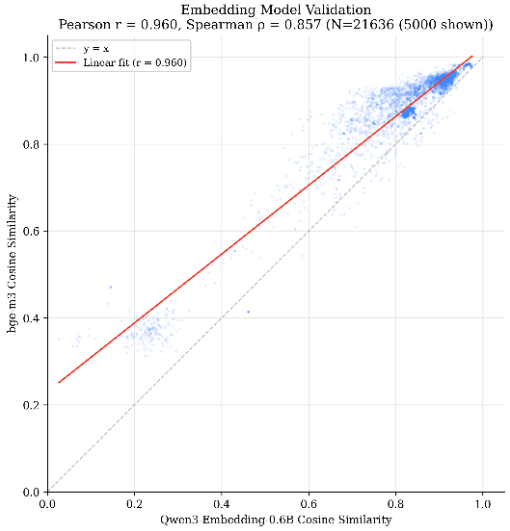

Figure 19: Embedding model validation on the cross-family data: cosine similarity scores from Qwen3-Embedding-0.6B vs. bge-m3 across all experimental conditions.

5 Discussion

Hierarchy of sensitivities. Read across the two datasets, the factors governing drift can be ordered by the size of the effect each produces on final similarity, with one qualification that the earlier single-dataset ordering did not need. Choice of model family remains the largest lever: 0.67 separates the best and worst cross-family models (0.904 vs. 0.239, Section 4.1). Within a family, parameter count matters up to the capacity threshold near 4B and little beyond it: clean-chain means for every configuration above the threshold lie within 0.10 of one another (Section 4.2). The next rung is not a continuous factor but a failure mode: runaway thinking removes 0–100% of a condition's chains depending on stack, precision, size, stimulus and prompt, and where it occurs it dominates every other effect (Section 4.3). Quantization format follows, at up to 0.26 between 4-bit builds of identical weights on the short stimulus and under 0.05 on the long one (Section 4.4). Temperature and KV-cache precision are the smallest continuous factors for stable configurations—within 0.06 across the temperature range for the stable cross-family models, and KV deltas of -0.10 to $+0.26$ of which eight of ten have intervals including zero—but below the threshold low temperature turns into a runaway-rate effect (Sections 4.5 and 4.6). System prompts are the qualification. On the most stable model they are the smallest factor of all (spread 0.008 on the short recipe, 0.054 on the long one), which is the ordering the cross-family dataset alone suggests; on every within-family configuration they are among the largest, with spreads of 0.3–0.7 and individual prompts able to trigger runaway in every chain. The prompt effect is thus not a fixed rung but a multiplier on how far a configuration already drifts. The practical reading is unchanged in direction and sharpened in emphasis: choose the model family first, stay above the capacity threshold, benchmark the exact quantized build on the exact stack, check explicitly for runaway and empty completions, and only then tune prompts—knowing that a prompt tuned on one build may be the worst choice on another.

Fixed-point attractors. We observe that a substantial fraction of runs converge to fixed points where similarity remains constant for 10+ consecutive iterations. This is consistent with the iterated function system formalization of Chytas and Singh [2026] and the attractor cycle dynamics of Wang et al. [2025]. Models with strong instruction-tuning (e.g., qwen3:30b-instruct) appear to have stable attractors near the original meaning, while weaker models' attractors lie far from the input distribution. The inverted temperature profiles of Section 4.6 are a different phenomenon: at $T \leq 0.3$ the small and 4-bit reasoning-enabled configurations do not lock onto an attractor but run away inside the thinking block, and added sampling entropy is what lets the reasoning terminate.

Cross-model rank inversion. The near-zero Spearman correlation between 30B and 8B prompt rankings (Section 4.7) likely reflects an interaction between prompt complexity and instruction-following capacity [Jaroslawicz et al., 2025]. The 30B model has sufficient capacity to simultaneously obey negative constraints ("never reference the chain") and paraphrase faithfully, so anti-drift prompts excel. The 8B model, with less headroom, appears to trade off constraint adherence against paraphrase quality: lightweight prompts like Natural & Engaging outperform because they do not compete for the model's limited instruction-following bandwidth. The within-family results show that this is not only a size effect: rankings also diverge between quantization formats of the same checkpoint, where capacity is nominally unchanged.

Non-monotonic precision effects. The finding that fp16 KV cache can *worsen* drift in some models, and that BF16 weights can lose to Q8_0 or Q4_K_M at the same size, challenges the

assumption that higher precision is always better for sequential generation. We hypothesize that quantization noise can act as a beneficial regularizer, preventing the model from precisely tracking latent attractor gradients. This parallels findings in noise injection for optimization and is consistent with Han et al. [2026]'s finding that quantization errors compound across sequential reasoning steps. At 4-bit, however, the dominant risk changes character: what limits the Q4_K_M configurations is not gradual fidelity loss but the discrete generation failure of Section 4.3.

Practical recommendations. Based on our findings, we offer the following recommendations for multi-agent LLM system designers:

1. **Model selection is paramount.** Choose instruction-tuned models with demonstrated stability; model size alone does not predict drift resistance.
2. **Stay above the capacity threshold.** Models below roughly 4B parameters are unreliable for iterated tasks regardless of quantization.
3. **Benchmark the exact quantized build.** GGUF, AWQ, NVFP4 and GPTQ-Int4 are not interchangeable for drift-sensitive applications even at identical nominal precision; test all candidate formats on representative tasks.
4. **Do not expect KV-cache precision to rescue drift.** Its per-model effects were small and mixed in sign, with eight of ten intervals including zero; treat it as a throughput choice and verify it does no harm on your build.
5. **Test at operating temperature.** Drift behavior changes non-linearly with temperature, and the direction of the effect inverts below the stability threshold; validate at the exact temperature used in production.
6. **Monitor for empty and runaway outputs explicitly.** Aggregate similarity scores hide the discrete failure modes of Section 4.3; check for empty completions and for unbounded length growth directly.
7. **Prompt complexity matters.** System prompts only help above a task complexity threshold; invest in prompt engineering for complex pipelines, not simple ones.
8. **Prompts don't transfer.** Within the twenty prompts tested, optimal system prompts were specific to the deployed configuration; re-evaluate when changing model, size or quantization format.
9. **Monitor for tipping points.** Catastrophic drift can occur suddenly; implement similarity monitoring with early stopping.
10. **Use multiple seeds for evaluation.** Single-run evaluations can be misleading; our 8B model shows final similarity ranges of 0.41–0.92 across seeds for the same prompt.

Limitations.

- **Single domain and single task.** All stimuli are cooking recipes and all chains use one paraphrase instruction; other domains and other iterated tasks (summarization, translation) may drift differently.

- **Serving stacks and samplers.** Results cover Ollama, vLLM and SGLang on one lab's hardware. In three of the four quantization-format arms the format and the backend covary, so the format effect is a format-and-stack effect. The vLLM arms were launched with thinking disabled and the Ollama and SGLang arms with it enabled, so the absence of runaway thinking on vLLM is a reasoning-off result, not a stack result. Sampling parameters other than temperature were left at each server's defaults: Ollama applied the library tags' presence penalty of 1.5, SGLang applied none. The companion paper shows that this penalty is the parameter that governs runaway, so the Ollama–SGLang gap is a sampler effect and was not controlled by design.

- **Single metric.** Cosine similarity captures one dimension of drift. Belem et al. [2026] show that repeated rewriting shifts expressed certainty while leaving surface content intact, a change this metric cannot see; the drift reported here is a lower bound on total meaning change, and the validation against a second embedding model (Section 4.8) bounds only the metric's consistency, not its coverage.

- **Seeds are samples, not replicates.** Fixing the sampling seed does not pin a chain on these stacks: under batched inference, identical seeds produce different trajectories because batch composition and kernel scheduling vary between requests [He and Thinking Machines Lab, 2025]. Every "seeded" run is one independent sample, every interval is an interval over those samples, and the Kruskal–Wallis and Wilcoxon statistics, which treat chains as observations, are reported as descriptive rather than inferential.

- **Small n where it matters most.** Several of the sharpest contrasts rest on two or three clean chains, because the same conditions that separate the formats are the ones that lose chains to runaway. The 35B Q4_K_M and NVFP4 clean means, in particular, should be read as existence results until replicated at higher n.

- **Several prompts are written for the three-step stimulus.** Six of the twenty system prompts (Deterministic Relay, 100% Fidelity, Numbered-Only Strict, Natural & Engaging, Zero Fluff and Locked Format) refer to "three steps" or prescribe a "Step 1 ... Step 3" format. They were nonetheless run unchanged on the long recipe, where those instructions contradict the input. Their lasagna rankings therefore measure a prompt that is partly self-contradictory, and the complexity-threshold comparison inherits that confound; the companion paper uses only the domain-neutral prompts when it moves beyond recipes.

- **The lasagna stimulus is compressed.** The 21-step recipe sent to the models collapses steps 12–17 into one line (Appendix A); it is a 16-line stimulus that nominally spans 21 steps.

- **Incomplete conditions.** Eleven of the 669 cross-family chains and 38 of the 2,007 within-family chains did not reach iteration 30. They are reported at their actual n or excluded as stated in Section 3.4; three of the serving endpoints involved have since been retired and those conditions cannot be regenerated on the same stack.

- **One out-of-family control.** The Nemotron-3-Nano-30B arm is a single model and cannot by itself establish how prompt rankings behave across families; it shows only that they need not transfer.

- **No quantization-aware training.** All quantized builds use post-training quantization.

Future work. Extending this framework to non-recipe domains (code generation, legal documents, medical instructions) would test the generality of the tier hierarchy and attractor dynamics observed here. Complementary metrics beyond cosine similarity—such as exact-match entity and quantity tracking, BERTScore, expressed-certainty measures in the manner of Belem et al. [2026], or LLM-as-a-judge evaluation for factual preservation—would capture dimensions of drift (e.g., hallucinated details, unit substitutions) that embedding-space distance may miss. A companion study examines the determinism, decoding and cross-generation questions that this one brackets.

6 Conclusion

We presented a systematic study of semantic drift in iterated LLM paraphrase chains, combining a cross-family sweep of 17 models with a controlled within-family sweep of the Qwen3.5 family, and isolating model architecture, scale, quantization level and format, serving infrastructure, temperature and prompt engineering. Across 2,676 launched chains (2,627 completed) and roughly 79,000 scored inferences we established that:

1. Model family dominates every other factor, producing a stable four-tier stability hierarchy with catastrophic tipping-point failures.
2. Within a family, a stability threshold sits at approximately 4B parameters, below which drift behavior becomes unpredictable, and above which the ordering by size is irregular.
3. Quantization effects are non-monotonic and size-dependent, and four 4-bit formats applied to identical weights produce measurably different drift dynamics.
4. KV cache precision has non-monotonic, directionally mixed effects.
5. Drift is not reducible to sampling entropy; below the stability threshold, low temperature raises the runaway-thinking rate rather than changing paraphrase fidelity.
6. System prompts are effective only above a stimulus complexity threshold, and optimal prompts do not transfer across families, sizes, or quantization formats.

These findings have immediate implications for the design of reliable multi-agent LLM systems, where understanding and mitigating semantic drift is essential for maintaining output quality across sequential processing stages. The interactions between the factors are complex and non-monotonic, underscoring the need for empirical benchmarking of the specific model configuration to be deployed rather than of the family it belongs to.

Use of AI assistance

The majority of the experiment, analysis and application code in the accompanying repository was written by large language models—Claude Opus and Claude Sonnet (Anthropic) and Grok (xAI)—working under the author's direction, with the author specifying the experiments, reviewing the code and its outputs, and correcting errors. Claude Fable and Claude Opus, Grok, and Gemini (Google) were used to aid the writing and layout of this manuscript, including drafting, editing, reference checking and LaTeX production. The experimental design, the interpretation of results, the hardware and serving facts reported, and all claims made here are the author's responsibility.

References

Alberto Acerbi and Joseph M. Stubbersfield. Large language models show human-like content biases in transmission chain experiments. *Proceedings of the National Academy of Sciences*, 120(44): e2313790120, 2023. doi: 10.1073/pnas.2313790120.

Catarina G. Belem, Shang Wu, Hongyu Yao, Mark Steyvers, Sameer Singh, and Padhraic Smyth. From 'may' to 'is': Certainty distortion in language model rewriting. *arXiv preprint arXiv:2606.07951*, 2026.

Levin Brinkmann, Fabian Baumann, Jean-François Bonnefon, Maxime Derex, Thomas F. Müller, Anne-Marie Nussberger, Agnieszka Czaplicka, Alberto Acerbi, Thomas L. Griffiths, Joseph Henrich, Joel Z. Leibo, Richard McElreath, Pierre-Yves Oudeyer, Jonathan Stray, and Iyad Rahwan. Machine culture. *Nature Human Behaviour*, 7(11):1855–1868, 2023. doi: 10.1038/s41562-023-01742-2.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216*, 2024.

Sotirios Panagiotis Chytas and Vikas Singh. Concept attractors in LLMs and their applications. *arXiv preprint arXiv:2601.11575*, 2026. Submitted 30 December 2025.

Olive Jean Dunn. Multiple comparisons using rank sums. *Technometrics*, 6(3):241–252, 1964. doi: 10.1080/00401706.1964.10490181.

Bradley Efron and Robert J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall/CRC, New York, 1993. doi: 10.1007/978-1-4899-4541-9.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. GPTQ: Accurate post-training quantization for generative pre-trained transformers. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.17323.

Georgi Gerganov and llama.cpp contributors. llama.cpp: LLM inference in C/C++. `https://github.com/ggml-org/llama.cpp`, 2023. Software repository; source of the GGUF format and the K-quant schemes (Q4_K_M, Q8_0).

Henry Han, Xiyang Liu, Xiaodong Wang, Fei Han, and Xiaodong Li. The quantization trap: Breaking linear scaling laws in multi-hop reasoning. *arXiv preprint arXiv:2602.13595*, 2026.

Horace He and Thinking Machines Lab. Defeating nondeterminism in LLM inference. `https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/`, sep 2025. Connectionism blog, Thinking Machines Lab.

Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W. Mahoney, Yakun Sophia Shao, Kurt Keutzer, and Amir Gholami. KVQuant: Towards 10 million context length LLM inference with KV cache quantization. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2401.18079.

Daniel Jaroslawicz, Brendan Whiting, Parth Shah, and Karime Maamari. How many instructions can LLMs follow at once? *arXiv preprint arXiv:2507.11538*, 2025.

Simon Kirby, Hannah Cornish, and Kenny Smith. Cumulative cultural evolution in the laboratory: An experimental approach to the origins of structure in human language. *Proceedings of the National Academy of Sciences*, 105(31):10681–10686, 2008. doi: 10.1073/pnas.0707835105.

William H. Kruskal and W. Allen Wallis. Use of ranks in one-criterion variance analysis. *Journal of the American Statistical Association*, 47(260):583–621, 1952. doi: 10.1080/01621459.1952.10483441.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP)*, pages 611–626, 2023. doi: 10.1145/3600006.3613165.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. AWQ: Activation-aware weight quantization for on-device LLM compression and acceleration. In *Proceedings of Machine Learning and Systems (MLSys)*, 2024. arXiv:2306.00978.

Zirui Liu, Jiayi Yuan, Hongye Jin, Shaochen Zhong, Zhaozhuo Xu, Vladimir Braverman, Beidi Chen, and Xia Hu. KIVI: A tuning-free asymmetric 2bit quantization for KV cache. In *International Conference on Machine Learning (ICML)*, 2024. arXiv:2402.02750.

Amr Mohamed, Mingmeng Geng, Michalis Vazirgiannis, and Guokan Shang. LLM as a broken telephone: Iterative generation distorts information. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025. arXiv:2502.20258.

Sabbir Mollah, Rohit Gupta, Sirnam Swetha, Qingyang Liu, Ahnaf Munir, and Mubarak Shah. The telephone game: Evaluating semantic drift in unified models. *arXiv preprint arXiv:2509.04438*, 2025.

NVIDIA, Felix Abecassis, Anjulie Agrusa, Dong Ahn, Jonah Alben, et al. Pretraining large language models with NVFP4. *arXiv preprint arXiv:2509.25149*, 2025.

Ollama. Ollama. `https://ollama.com`, 2026. Software; local LLM serving runtime built on llama.cpp. Version 0.16.3 was used in this work.

Jérémy Perez, Grgur Kovač, Corentin Léger, Cédric Colas, Gaia Molinaro, Maxime Derex, Pierre-Yves Oudeyer, and Clément Moulin-Frier. When LLMs play the telephone game: Cultural attractors as conceptual tools to evaluate LLMs in multi-turn settings. In *International Conference on Learning Representations (ICLR)*, 2025. arXiv:2407.04503.

Qwen Team. Qwen3.5-35b-a3b model card. `https://huggingface.co/Qwen/Qwen3.5-35B-A3B`, 2026. Hugging Face model card, created 2026-02-24, Apache-2.0 license; the card links to the Qwen3.5 release blog post.

Abhishek Rath. Agent drift: Quantifying behavioral degradation in multi-agent LLM systems over extended interactions. *arXiv preprint arXiv:2601.04170*, 2026.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Yarin Gal, Nicolas Papernot, and Ross Anderson. The curse of recursion: Training on generated data makes models forget. *arXiv preprint arXiv:2305.17493*, 2023.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759, 2024. doi: 10.1038/s41586-024-07566-y.

James H. Smith. Semantic drift across Qwen generations: Reproducibility, thinking mode, serving stack, and what cosine similarity misses in iterated paraphrase chains. Preprint, Zenodo, 2026a. Companion paper, referred to as Paper B.

James H. Smith. System prompt engineering and semantic drift in multi-agent LLM pipelines. `https://joshua8.ai/system-prompt-engineering-semantic-drift/`, 2026b. Blog post, Joshua.AI LLC.

Zhilin Wang, Yafu Li, Jianhao Yan, Yu Cheng, and Yue Zhang. Unveiling attractor cycles in large language models: A dynamical systems view of successive paraphrasing. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12740–12755, Vienna, Austria, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.acl-long.624. arXiv:2502.15208.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*, 2025.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024. arXiv:2312.07104.

A Prompt Texts

Spaghetti (3 steps). The stimulus is reproduced verbatim from `PROMPTS["spaghetti"]` in the experiment scripts:

> Step 1: Boil water. Step 2: Add pasta for 8 minutes. Step 3: Drain and serve with sauce.

Lasagna (nominally 21 steps). The text below is the verbatim stimulus sent to the models (`PROMPTS["lasagna"]` in the same scripts). Note that although the recipe is numbered through step 21, steps 12–17 are compressed into a single line in the stimulus itself, so the transmitted text contains 16 lines rather than 21. We reproduce it exactly rather than expanding it, because the compression is part of the condition being measured.

> Follow these exact steps to prepare a classic homemade beef lasagna that serves 8–10 people:
>
> Step 1: Preheat your oven to 375°F (190°C) and lightly grease a 9×13 inch baking dish. Step 2: Bring a large pot of lightly salted water to a boil. Step 3: Heat 2 tablespoons of olive oil in a large skillet over medium heat. Step 4: Add 1 large finely diced onion and cook for 5 minutes until translucent. Step 5: Add 5 minced garlic cloves and cook for 1 minute until fragrant. Step 6: Add 1.25 lbs ground beef and 0.75 lb Italian sausage. Cook until browned. Step 7: Drain excess grease from the skillet. Step 8: Stir in two 28-oz cans of crushed tomatoes, 3 tbsp tomato paste, 2 tsp dried basil, 1.5 tsp dried oregano, 0.5 tsp fennel seed, 1 tsp sugar, 1 tsp salt, and 0.5 tsp black pepper. Step 9: Bring to a simmer and cook the meat sauce for 30 minutes. Step 10: Combine 15 oz ricotta, 1 egg, 1/2 cup Parmesan, 1/4 cup parsley, 1/2 tsp salt, 1/4 tsp pepper. Step 11: Cook 12 lasagna noodles until al dente. Drain and rinse. Step 12–17: Layer sauce, noodles, ricotta, mozzarella (3 layers). Step 18: Cover with foil and bake 25 minutes. Step 19: Remove foil and bake 25–30 more minutes until golden. Step 20: Rest 15–20 minutes before slicing. Step 21: Garnish with fresh basil and serve hot.

System prompts (20 total). The following system prompts span 6 categories: Baseline (2), Anti-Drift (6), Constrained (4), Creative (4), VL-Leveraged (3), and Thinking (1).

Baseline

1. Baseline – None. *(empty string)*

2. Minimal Helpful. "You are a helpful assistant."

Anti-Drift

3. Strong Anti-Meta. "You are a precise recipe transmitter in a multi-agent kitchen pipeline. Your only job is to paraphrase the incoming message while preserving every single step and detail exactly. Never add, remove, or rephrase any instruction. Never add meta-commentary, explanations, or agent framing. Output only the paraphrased steps."

4. Ultra-Constrained Fidelity. "Maintain exact semantic meaning at all costs. Preserve every numbered step verbatim where possible. Never add fluff, commentary, or new information. Respond with nothing but the updated numbered list."

7. 100% Fidelity. "Preserve 100% of the original meaning and all three steps. Do not omit, alter, or add any detail. Never include meta-commentary."

12. Precise Transmitter. "You are a precise recipe transmitter. Paraphrase the message without losing or adding any information. Stay strictly factual and neutral."

16. Multi-Agent Relay. "You are part of a multi-agent recipe relay system. Pass the message forward with improved clarity but zero loss of meaning. Never reference the chain or agents."

19. No Meta Ever. "Do not mention agents, pipelines, paraphrasing, instructions, or the task itself in any way. Just output the rephrased recipe steps cleanly."

Constrained

6. Deterministic Relay. "You are a deterministic message relay. Re-express the input using different words but identical semantics and structure. Output format must be exactly: Step 1: ... Step 2: ... Step 3: ... Nothing else."

8. Numbered-Only Strict. "Output ONLY a numbered list of the three steps. No introductions, conclusions, explanations, or extra text of any kind."

13. Zero Fluff. "Never add fluff, commentary, explanations, or new information. Stick strictly to rephrasing the existing three steps only."

14. Locked Format. "Always respond in this exact format and nothing else: Step 1: [rephrased step] Step 2: [rephrased step] Step 3: [rephrased step]"

Creative

9. Natural & Engaging. "Make this recipe message more natural and engaging for the next team member while preserving 100% of the original meaning and all three steps. Keep the same numbered format. No introductions or conclusions."

10. Progressive Refinement. "You are part of a long-running chain of agents. Each paraphrase must be slightly clearer and easier to follow than the last while staying perfectly faithful to the original intent. Avoid repetition and any meta language."

15. Friendly Conversational. "Rephrase the recipe in a friendly, conversational tone while keeping all steps and details 100% accurate. Make it sound like helpful advice from one cook to another."

20. Balanced Clarity. "Balance natural, readable language with perfect fidelity. Make the instructions clearer and more readable for the next person without changing any content or adding new ideas."

VL-Leveraged

5. Expert VL Cooking Assistant. "You are an expert visual cooking assistant. Paraphrase the recipe steps clearly and naturally while keeping every action intact. Use vivid but precise language that would work well in a visual tutorial. Do not add meta-notes or agent talk."

11. Visual Tutorial Style. "As a visual cooking expert, paraphrase the steps in a way that would pair perfectly with images or a cooking video. Keep every step intact and use language that helps someone visualize the actions."

18. Multimodal Visual Focus. "You are a multimodal cooking assistant trained on thousands of visual recipes. Paraphrase the steps to make them extremely easy to visualize and follow in a real kitchen."

Thinking

17. Think-Step-by-Step. "Think step by step about the meaning of each instruction, then output only the final paraphrased numbered steps. Do not show your thinking."

B Model Inventory

Cross-family configurations. Table 8 lists every served model configuration used in the cross-family experiments, with the parameter count and quantization parsed from the served model tag, the serving backend, and the experiments each configuration took part in. The table is generated directly from the configuration block recorded inside each experiment's raw data file, so it reflects exactly what was served rather than a hand-maintained list. Where the model tag does not state a parameter count or a quantization (for example a Hugging Face repository name without a size suffix), the field is shown as "—" rather than inferred.

Within-family configurations. Table 9 summarizes the 22 Qwen3.5 configurations tested together with the out-of-family control. All GGUF models were served with Ollama. The vLLM models were `cyankiwi/Qwen3.5-35B-A3B-AWQ-4bit` (AWQ 4-bit), `Qwen/Qwen3.5-35B-A3B-GPTQ-Int4`

Table 8: Complete model inventory, generated from the `models_config` block recorded in each experiment's raw data file. Parameter counts and quantization are parsed from the served model tag; "—" means the tag does not state that field. "Temp" is the temperature ablation of Section 4.6.

Served model tag	Params	Quantization	Backend	Experiments
`qwen3:0.6b-fp16`	0.6B	fp16	Ollama	RQ2
`qwen3:0.6b-q4_K_M`	0.6B	Q4_K_M	Ollama	RQ2
`qwen3:1.7b-q8_0`	1.7B	Q8_0	Ollama	RQ2
`qwen3:4b-instruct-2507-fp16`	4B	fp16	Ollama	RQ2
`qwen3:4b-instruct-2507-q4_K_M`	4B	Q4_K_M	Ollama	RQ2
`llama3.1:8b`	8B	—	Ollama	RQ1, Temp
`qwen3:8b-q4_K_M`	8B	Q4_K_M	Ollama	RQ2, RQ3, Temp
`ministral-3:14b`	14B	—	Ollama	RQ1
`devstral-small-2:24b`	24B	—	Ollama	RQ1
`gemma3:27b-it-qat`	27B	QAT	Ollama	RQ1, Temp
`hf.co/mradermacher/medgemma-27b-multimodal-GGUF:latest`	27B	—	Ollama	RQ1
`cpatonn/Qwen3-VL-30B-A3B-Instruct-AWQ-4bit`	30B	AWQ 4-bit	vLLM	RQ1, RQ3
`nemotron-3-nano:30b`	30B	—	Ollama	RQ1
`qwen3-coder:30b`	30B	—	Ollama	RQ1
`qwen3-vl:30b-a3b-instruct-q4_K_M`	30B	Q4_K_M	Ollama	RQ1
`qwen3:30b-a3b-instruct-2507-q4_K_M`	30B	Q4_K_M	Ollama	RQ1, RQ2, Temp
`qwen3:30b-a3b-instruct-2507-q8_0`	30B	Q8_0	Ollama	RQ2
`qwen3:30b-a3b-thinking-2507-q4_K_M`	30B	Q4_K_M	Ollama	RQ1, RQ2
`qwen3-next:80b-a3b-instruct-q4_K_M`	80B	Q4_K_M	Ollama	RQ1, RQ2
`gpt-oss:120b`	120B	—	Ollama	RQ1
`devstral-2:123b`	123B	—	Ollama	RQ1
`glm-4.7-flash:Q4_K_M`	—	Q4_K_M	Ollama	RQ1
`hf.co/bartowski/zai-org_GLM-4.5-Air-GGUF:latest`	—	—	Ollama	RQ1
`qwen3-coder-next:Q4_K_M`	—	Q4_K_M	Ollama	RQ1

(GPTQ-Int4) and `cyankiwi/Qwen3.5-9B-AWQ-4bit` (9B AWQ) and, as the out-of-family control, `nvidia/NVIDIA-Nemotron-3-Nano-30B-A3B-NVFP4`. The SGLang model was `txn545/Qwen3.5-35B-A3B-NVFP4`.

Table 9: Within-family (Qwen3.5) model inventory. GGUF models were served with Ollama; the AWQ and GPTQ-Int4 variants and the Nemotron control with vLLM; the Qwen3.5 NVFP4 variant with SGLang. The Nemotron-3-Nano-30B row is an out-of-family control, included to test whether the system prompt rankings of Section 4.7 are specific to the Qwen3.5 family. See Appendix E for hardware and software versions.

Size	Architecture	Quantizations	Backend
0.8B	Dense	Q8_0, BF16	Ollama
2B	Dense	Q4_K_M, Q8_0, BF16	Ollama
4B	Dense	Q4_K_M, Q8_0, BF16	Ollama
9B	Dense	Q4_K_M, Q8_0, BF16	Ollama
27B	Dense	Q4_K_M, Q8_0, BF16	Ollama
35B	MoE (A3B)	Q4_K_M, Q8_0, BF16	Ollama
35B	MoE (A3B)	AWQ 4-bit	vLLM
35B	MoE (A3B)	NVFP4 4-bit	SGLang
35B	MoE (A3B)	GPTQ-Int4	vLLM
9B	Dense	AWQ 4-bit	vLLM
122B	MoE	Q4_K_M	Ollama
Out-of-family control			
30B	MoE (A3B)	NVFP4 4-bit	vLLM

Note: Qwen3.5 0.8B Q4_K_M does not exist as a released quantization. Qwen3.5 122B is only

available in Q4_K_M under our GPU memory constraints. No official GPTQ-Int4 build of Qwen3.5 9B exists, and the available 9B NVFP4 builds come from publishers other than the one used for the 35B NVFP4 model, so a 9B NVFP4 arm was excluded to avoid a calibration confound.

C Additional Figures

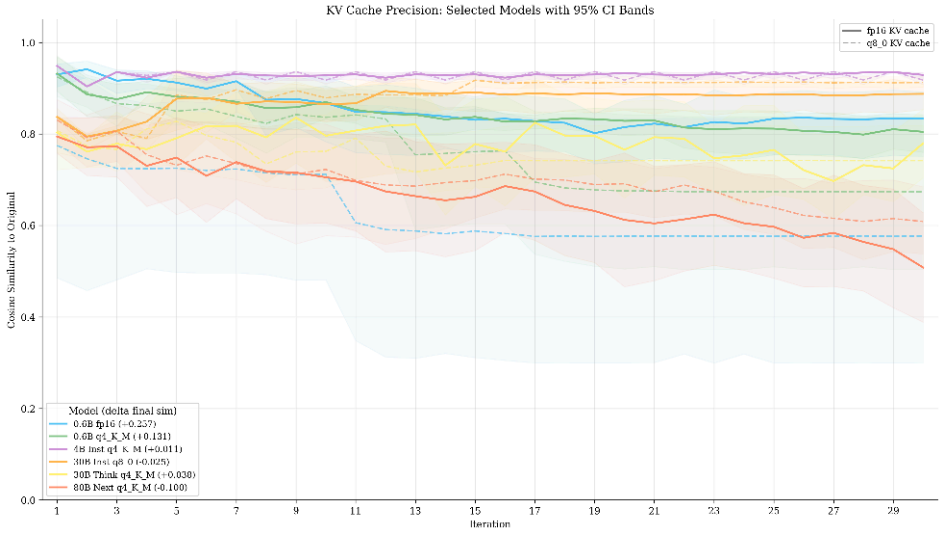

Figure 20: KV cache precision: selected models with 95% CI bands. Solid lines = fp16 KV cache; dashed = q8_0 KV cache. 5 seeds per condition.

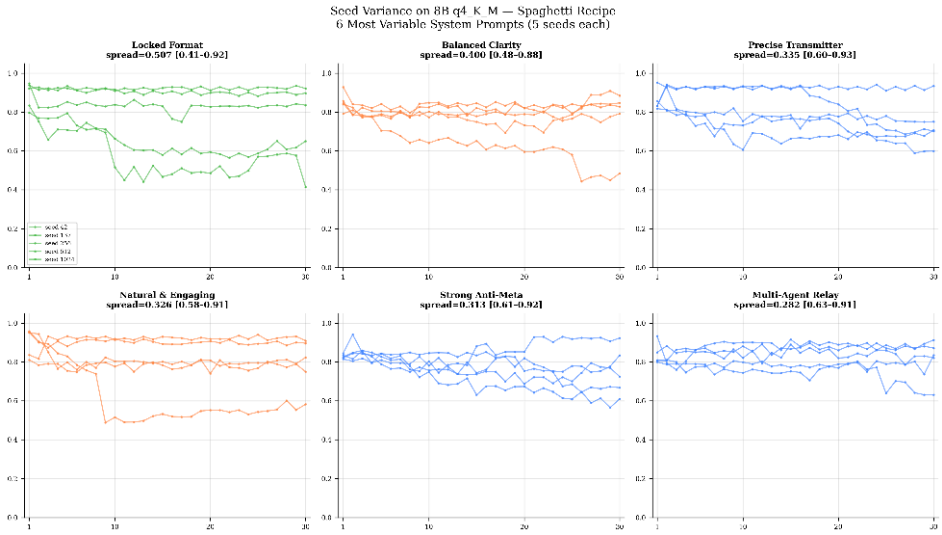

Figure 21: Seed variance on 8B q4_K_M, spaghetti recipe. The 6 most variable system prompts show dramatically different trajectories across 5 seeds.

D Embedding Model Validation Detail

We validate the primary embedding model (Qwen3-Embedding-0.6B, 32,768-token context) against bge-m3 (8,192-token context) by re-embedding all saved text pairs and computing Pearson and Spearman correlations between the two models' cosine similarity scores.

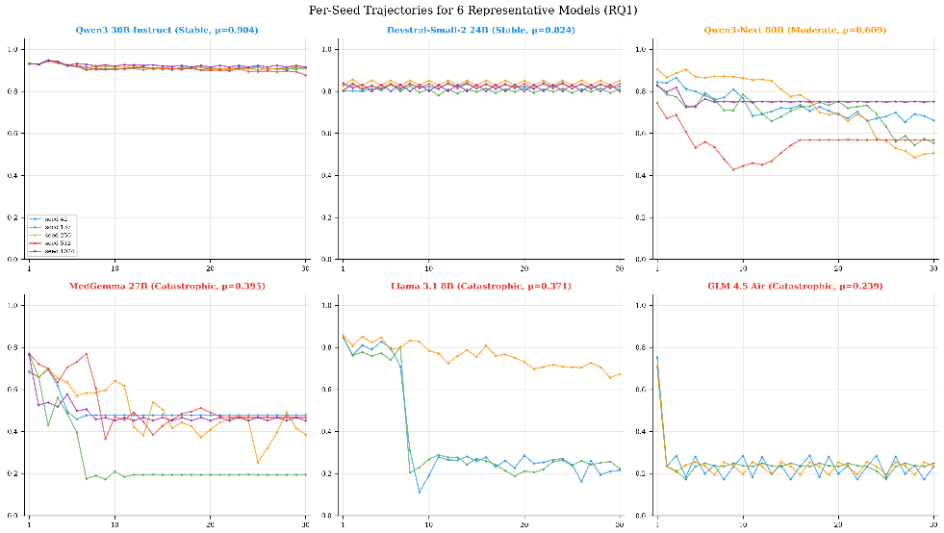

Figure 22: Per-seed trajectories for 6 representative cross-family models.

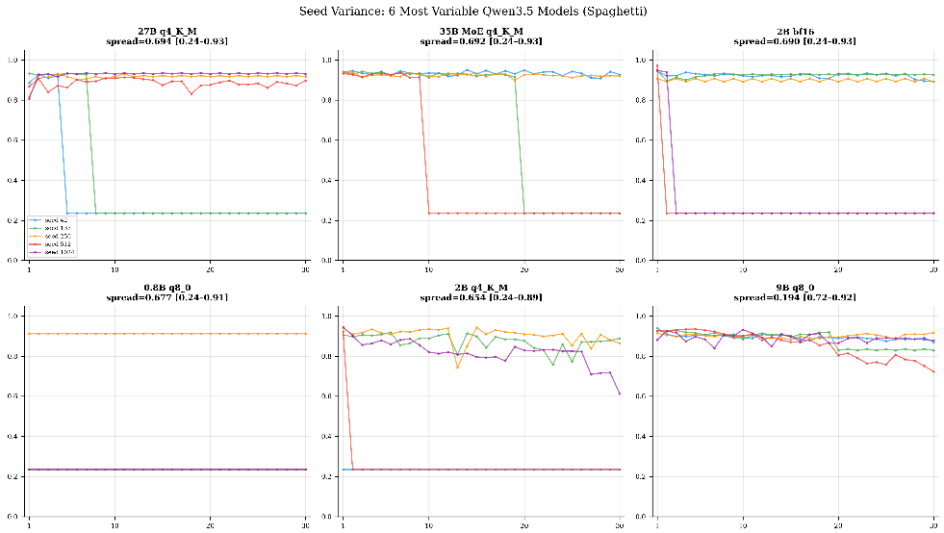

Figure 23: Seed variance for the 6 most variable Qwen3.5 models on the spaghetti recipe. Individual seed trajectories shown.

On the cross-family data, all 21,636 pairs embedded successfully, giving $r = 0.960$ and $\rho = 0.857$ (Figure 19).

On the within-family data, 20,072 of 22,167 pairs (90.6%) produced valid similarities from both models, yielding $r = 0.825$ and $\rho = 0.815$ ($p < 10^{-10}$ for both; Figure 25). The remaining 2,095 pairs (9.4%) failed because bge-m3's vLLM endpoint rejects inputs exceeding its 8,192-token context window rather than truncating them. All failed pairs are outputs exceeding 6,000 words, produced by chains that experienced runaway text expansion—a characteristic drift failure mode in which certain system prompts (e.g., "Visual Tutorial Style," "Friendly Conversational") cause outputs to grow by several thousand words over 30 iterations. These excluded pairs have a mean Qwen similarity of 0.515 (median 0.572) compared with the dataset mean of 0.829, confirming that they are systematically the most-drifted outputs. Because both embedding models would score

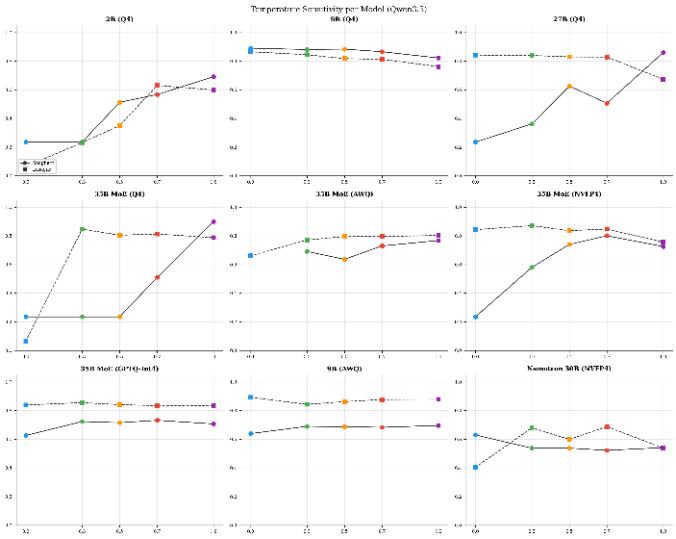

Figure 24: Temperature sensitivity per Qwen3.5 model: mean final similarity at each temperature for spaghetti (solid) and lasagna (dashed).

these highly-diverged texts as low similarity, their exclusion is unlikely to meaningfully affect the correlation.

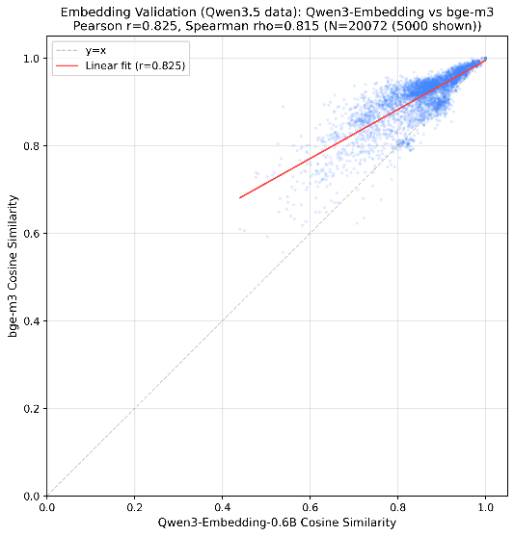

Figure 25: Embedding model validation scatter for the within-family data: Qwen3-Embedding-0.6B vs. bge-m3 cosine similarities for 20,072 Qwen3.5 experiment text pairs (2,095 pairs with outputs exceeding bge-m3's 8,192-token context limit excluded). Pearson $r = 0.825$, Spearman $\rho = 0.815$.

E Reproducibility and Compute

Hardware. All Ollama and vLLM runs were executed on a single on-premises server with one NVIDIA RTX PRO 6000 Blackwell GPU (96 GB), except that the two NVFP4 models—the 35B NVFP4 build on SGLang and the Nemotron control on vLLM—were executed on a second on-

premises server with one NVIDIA RTX 5090 GPU (32 GB). Embeddings were computed with Qwen3-Embedding-0.6B served by vLLM on a dedicated NVIDIA RTX 5070 Ti GPU (16 GB). The validation embeddings used bge-m3: for the cross-family dataset served by Ollama on the RTX PRO 6000, and for the within-family dataset served by vLLM on an RTX 5070 Ti.

Software. Ollama v0.16.3; vLLM v0.15.1, plus a CUDA 13.0 nightly build of vLLM for the GPTQ-Int4, 9B AWQ and Nemotron NVFP4 runs; SGLang v0.5.9 for the Qwen3.5 35B NVFP4 runs. Analysis used Python with NumPy, SciPy and Matplotlib, driven by `scripts/analyze_paper.py` and `scripts/analyze_qwen35_paper.py`, which regenerate every figure, table and reported statistic in this paper from the raw per-iteration records; dependencies are pinned with `uv`.

Scale and compute budget. The cross-family experiments comprise 669 seeded chains and 20,070 attempted inferences, of which 19,798 returned a scored output; 658 chains ran the full 30 iterations. Summing the recorded per-iteration wall-clock times across those data files gives approximately 61 hours of cumulative inference time. The within-family experiments comprise 1,969 completed 30-iteration chains, that is 59,070 inferences, drawn from 2,007 launched chains; summing their recorded per-iteration times gives approximately 630 hours of cumulative inference time. Because several of those experiments ran with up to four concurrent workers, the elapsed wall-clock duration of the campaign was substantially shorter than that figure. Embedding validation added 21,636 and 22,167 further embedding pairs respectively.

Availability. The experiment scripts, the analysis code that produces every figure and table in this paper, and the raw per-iteration drift records (every chain's input and output text) are available at `https://github.com/jhsmith409/telephone-semantic-drift`; the drift records are attached to the repository's data releases as compressed archives with published SHA-256 checksums. The same data, code and paper sources are archived at Zenodo: the dataset record has DOI 10.5281/zenodo.22646927, this paper 10.5281/zenodo.22646929, and the companion paper [Smith, 2026a] 10.5281/zenodo.22646931.