GACHA DECODING: ELICITING DIVERSE GENERATIONS THROUGH INSTRUCTION FOLLOWING

Anonymous authors
Paper under double-blind review

ABSTRACT

Language models often produce surprisingly homogeneous responses to open-ended prompts, with a model's diversity tending to decline as its capability increases. We introduce Gacha Decoding, an inference-time method for eliciting diverse language model generations. Across open-ended domains (in-the-wild chat, creative writing, planning for image generation, and protein design), Gacha Decoding significantly outperforms existing approaches in diversity at equal quality (up to 2.3x Vendi over the next-best prior approach), reaching the same number of high-quality modes with over an order of magnitude fewer samples (11.0x), and discovering novel modes that no other approach surfaces. Our key insight is to treat diversity as an instruction-following problem: rather than relying on the LM's token entropy, we combine its instruction-following capability with randomness from an external RNG tool to scalably identify and realize distinct modes of the response space. This approach of "planning with dice" enables Gacha to invert the long-observed tension between diversity and model capability. As the underlying LM becomes a better instruction follower, diversity under Gacha Decoding consistently improves—even as its entropy decreases and diversity under prior approaches declines. Together, our results highlight that instruction following, rather than token entropy alone, can serve as the engine driving response diversity.

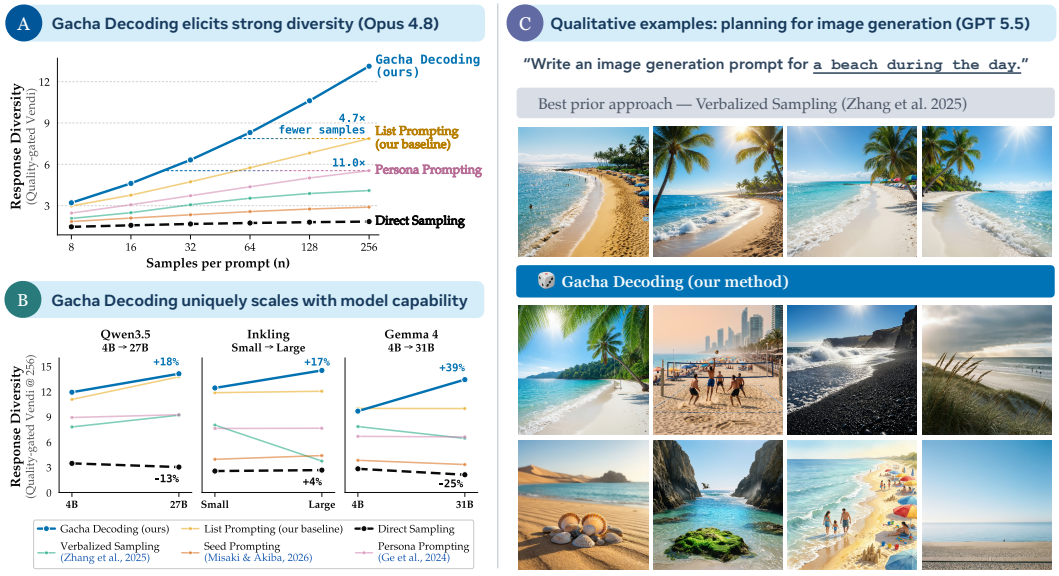


Figure 1: We introduce **Gacha Decoding**, an inference-time method that substantially improves LM generation diversity over prior approaches (**A**). We recast diversity as an instruction-following problem: instead of relying on token entropy for variation, Gacha leverages a model's IF capabilities along with external RNG to identify and realize distinct output modes. This reformulation enables Gacha to improve with stronger models (**B**), even as their token entropy decreases and diversity under prior approaches stagnates or declines. Panels A–B: results on InfinityChat (Jiang et al., 2026). Panel C: randomly selected images from text-to-image prompts generated by ours and baselines.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

1 INTRODUCTION

Language model (LM) systems are increasingly entrusted with creative or exploratory tasks (writing, ideation, data generation, and scientific discovery, to name a few), but they often lack the human-like diversity and novelty these applications demand. Modern models collapse onto just a few typical responses, even on open-ended prompts where we expect many possibilities (Jiang et al., 2026). Left unchecked, this shortfall lowers user utility (Zhang et al., 2025c), constrains practitioners who rely on diverse samples to improve synthetic data or scale test-time compute (Cui et al., 2025), and in the long run may risk homogenizing our thinking (Abdulhai et al., 2026; Yakura et al., 2024).

Spurred by this gap, researchers have made a flurry of attempts at improving generation diversity, often treating diversity as a problem of limited entropy in the LM's next-token distribution. Efforts have hence focused on broadening or reshaping this distribution, for example, by changing sampling parameters (Ahmed et al., 2025), applying entropy-encouraging losses (Shi et al., 2018; Li et al., 2025a; Sorensen et al., 2026), or curating diverse training data (Su et al., 2020; Zhang et al., 2024; Springer et al., 2026). While these approaches have shown gains, this distribution-centric paradigm of diversity has led to an apparent tension between response diversity and model capabilities, as capabilities are often improved via post-training methods that reduce entropy as a side effect (Cui et al., 2025). As models get better, diversity appears to be getting worse (Zhang et al., 2025c).

We argue that this tension is not fundamental. In this paper, we recast generation diversity as an instruction-following (IF) problem: we show that instruction following, as opposed to token entropy, can serve as the engine driving response diversity. In particular, we show how to harness a model's IF capabilities together with external randomness to scalably identify and realize distinct modes of the response space. This view crystallizes an emerging thread in the literature; indeed, many recent methods implicitly leverage instruction-following for diversity, for example by prompting to steer generation toward less typical responses (Zhang et al., 2026; Misaki & Akiba, 2026; Wu et al., 2025a; Zhang et al., 2025a). By treating diversity explicitly as an IF problem, we fundamentally shift the LM's role from maintaining diffuse distributions to faithfully executing instructions—a capability that can improve with better alignment, even if the token distribution becomes sharper.

We instantiate this view as **Gacha Decoding**[1], our inference-time method for eliciting diverse LM generations. Across open-ended chat, creative writing, planning for image generation, and protein design, **Gacha Decoding substantially outperforms prior best methods**, yielding up to a 2.3x gain in Vendi-scored diversity at similar or higher quality (Fig. 1A). Most strikingly, Gacha Decoding's diversity consistently scales with model capability, even when diversity with direct sampling or other prior approaches declines (Fig. 1B). Gacha thus inverts the typical diversity-capability tradeoff, reflecting its reduced reliance on token entropy. As an extreme illustration, Gacha's diversity gains persist even under greedy decoding, with every underlying LM call fully deterministic (Section 3.3).

How does Gacha Decoding work? At a high level, we "plan with dice" (Algorithm 1 and Fig. 2). Given a task, we instruct the LM to identify a sequence of semantic decisions that shape the output (e.g., to write a story, the setting, characters, and so on). We then sample these decisions sequentially using external RNG to pick from LM-generated options, and realize the resulting randomized plan into a final response. Crucially, we condition each decision on those already sampled so that each random choice reshapes the downstream options, allowing variation to compound combinatorially without token stochasticity. Overall, we view Gacha Decoding as autoregressive generation lifted from tokens to ideas: each step samples a semantic choice rather than a token, conditioned on all prior choices. In practice, it can be implemented as a lightweight harness around any existing LM.

Overall, our work engages an ongoing dialogue between (a) viewing LMs as classical generative models directly responsible for maintaining a diverse distribution, and (b) viewing them as components of larger AI systems. In a system, variation need not be encoded in the LM's distribution: we can source randomness externally and harness the LM's IF ability to propagate it into useful, diverse responses. **Our conceptual contribution** is to crystallize the view that generation diversity can be treated as an instruction-following problem. **Our technical contribution** is to show how planning with dice operationalizes this principle, yielding strong empirical results. Our inference-time results also suggest new training directions, including end-to-end optimization with our harness or distillation of our diverse generations back into the underlying model. We now detail Gacha Decoding (Section 2), compare it against prior methods (Section 3), and analyze why it works (Section 4).

[1] Named after *Gachapon* vending machines that dispense random toys from a large curated pool.

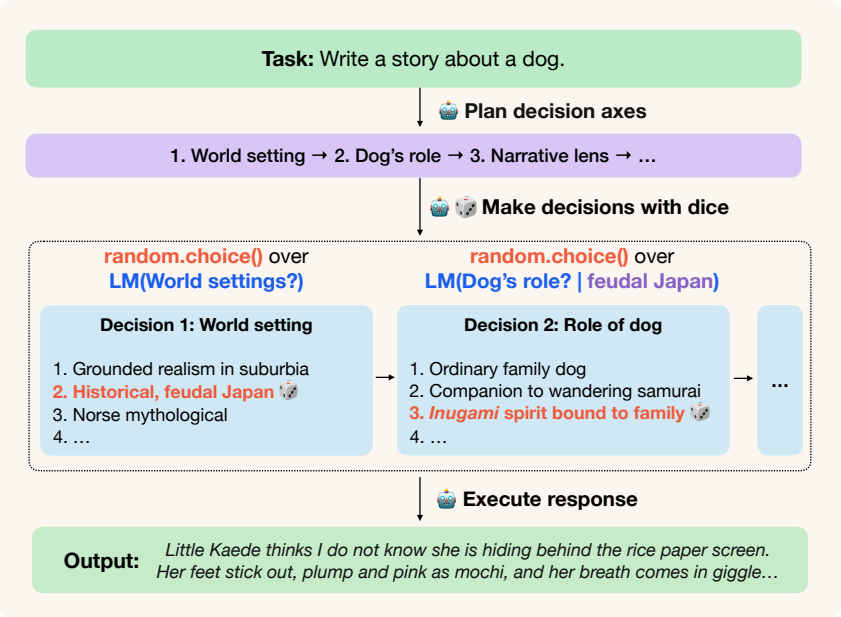


Figure 2: Gacha Decoding generates responses by "planning with dice." Given a task, we instruct the LM to identify and make a sequence of decisions to determine the final response. At each step, the LM enumerates possible options, and uses external randomness to select one uniformly. This design utilizes the LM's IF capabilities while minimizing reliance on token entropy.

2 Gacha Decoding: Diversity Through Instruction Following

2.1 Overview: What is Gacha Decoding?

We present Gacha Decoding, our method for eliciting diverse generations from language models. We first describe how Gacha Decoding works, and then explain the three key design elements underlying its success. We show the core algorithm in Algorithm 1 and a concrete example in Fig. 2. To summarize, given a task, the LM identifies a sequence of decisions that shape the output, then samples each decision by (1) enumerating plausible options and (2) selecting one with external RNG. For example, to write a story about a dog, we might first decide the setting, using the LM to list distinct options (e.g. modern suburbia, historical Japan, etc.) and picking one with dice. Later decisions—such as the dog's role or narrative perspective—are conditioned on that sampled choice, allowing diversity to compound across the decision trajectory. By using instruction following to enumerate options and external randomness to select among them, Gacha shifts the source of diversity away from token entropy. Overall, as we will detail more below, we can view this procedure as a randomized autoregressive process in idea-space. In practice, Gacha Decoding can be implemented as a simple agentic harness over an existing LM; see Appendix B for example prompts.

2.2 Diversity From Instruction Following by Planning with Dice

We explain the three design choices that shift Gacha Decoding's source of diversity from token entropy to instruction following: (1) work in semantic idea space, (2) sample ideas by decoupling enumeration from selection, and (3) sample ideas autoregressively. As we later show in our ablations (Section 4), each component is necessary. Now, to motivate these designs conceptually, let us build them up one at a time, starting from the simplest baseline: direct sampling from an LM. Suppose we are given a prompt x and generate a response y using standard token autoregression from an LM p_θ. Here, y is built as a sequence of stochastic token decisions, $p(y \mid x) = \prod_{i=1}^{T} p_\theta(y_i \mid x, y_{<i})$.

Working in idea space. Standard token sampling has two limitations for diverse generation. First, diversity fully relies on entropy in the LM's token distribution, which often becomes increasingly concentrated as models improve. Second, token variation is only a proxy for the semantic variation we care about: different token sequences may express the same idea, while unique approaches may

Algorithm 1 Gacha Decoding: diversity through instruction following by planning with dice

Require: Prompt x; language model LM; decision budget T; external RNG
Ensure: A diverse response y

Phase 1: Plan the decision sequence.

1: Instruct the LM to identify T semantic decision axes $(a_1, \ldots, a_T)$ that will shape the response. At this stage the model specifies only *which* decisions will be made, not their values.

Phase 2: Autoregressively sample the decisions.

2: **for** $t = 1, \ldots, T$ **do**
3: **Enumerate options.** Instruct the LM to list a set $\mathcal{C}_t$ of plausible, semantically distinct options for decision axis a_t, conditioned on the prompt and all decisions committed so far:
$$\mathcal{C}_t \leftarrow \mathrm{LM}(x,\, a_t,\, z_{<t})$$
4: **Roll dice.** Sample one option uniformly with external RNG and commit it to the trajectory:
$$z_t \sim \mathrm{Uniform}(\mathcal{C}_t)$$
5: **end for**

Phase 3: Realize the response.

6: **Integrate.** Integrate all decisions $z_{1:T}$ into a final plan: instruct the LM to propose several full plans, and select one uniformly with external RNG.
7: **Generate.** Instruct the LM to write the final response y from the prompt, the decision trajectory, and the selected plan, so that y is consistent with every sampled choice:
$$y \leftarrow \mathrm{LM}(x,\, z_{1:T},\, \text{plan})$$
8: **return** final response y

pivot at only a few key tokens. A natural first step is then to generate in two stages, re-factorizing generation to separate a response's high-level semantics from its surface realization. Formally, we introduce a latent variable z, denoting a high-level idea or plan for the response, and write

$$p(y \mid x) = \int p_\theta(y \mid z, x)\, q(z \mid x)\, dz. \tag{1}$$

Under this view, diverse generation reduces to sampling diverse semantic plans z, which the LM then realizes as responses. Prior methods often instantiate this idea by prompting an LM to generate the plan z directly (Yao et al., 2019; Ahmed et al., 2025; Springer et al., 2026), for example,

$$q(z \mid x) \approx p_\theta\left(z \mid \texttt{Give a plan for } x\right). \tag{2}$$

This moves variation to the semantic level, but does not yet change its source: diversity over plans again hinges on the LM's (limited) token entropy. We therefore seek a different way to sample z.

Decoupling idea enumeration from selection. Rather than using the LM to stochastically generate an idea directly, we use it to *enumerate* a set $\mathcal{C}$ of plausible, distinct choices and delegate *selection* to an external RNG, for example $z \sim \mathrm{Uniform}(\mathcal{C})$. This distinction is critical. While direct sampling relies on the LM assigning sufficient probability mass across diverse ideas, enumerating distinct ideas in-context is largely an instruction-following problem. Even under greedy decoding, a capable model can condition on what it has already generated and produce something new. We can view this random enumeration-selection process as sampling from $q(z \mid x)$ at very high effective temperature, where the support is verbalized by the LM. Variation over z then depends only on which possibilities the LM can enumerate, which can improve with IF capability. Working in idea space makes this disentanglement especially natural: we can randomize aggressively over semantic choices, such as a story setting or solution strategy, without sacrificing local token coherence.

Sampling ideas autoregressively. One could stop here and simply instruct the model to list complete plans z and randomly select among them. However, our diversity would be bottlenecked by the model's effective context window—when generating any list, the model can only output so many distinct options before degrading into low-quality or redundant options. Naively generating many lists with independent LM calls (e.g., Zhang et al. (2025a) generate parallel lists of full responses y) does not solve this problem because variation across lists would again depend on token stochasticity. To scale beyond this limitation, we make the idea sampling autoregressive. We factorize the

distribution over ideas $q(z \mid x)$ into a sequence of conditional semantic choices:

$$q(z \mid x) = q(z_{1:T} \mid x) = \prod_{t=1}^{T} q(z_t \mid x, z_{<t}). \tag{3}$$

Now, rather than using our LM to enumerate complete plans (i.e., from the joint distribution), we enumerate and randomize the next local choice conditioned on previous ones. With this autoregressive structure, each randomized choice changes the context for future LM enumeration calls, allowing us to explore combinatorially many trajectories even with a fixed number of choices at each step. Further, earlier choices reveal possibilities that are hard to anticipate in an unconditioned list. For example, once a story is placed in ancient Rome, subsequent decisions might involve gladiators or senate conspiracies, setting-specific elements that might never surface when listing complete story plans from scratch. Taken together, our three design choices yield Gacha Decoding, which samples diverse responses by autoregressively sampling diverse semantic choices using external randomness.

3 Main Experiments

3.1 Experimental Setup

We now compare Gacha Decoding against existing methods on three open-ended domains: in-the-wild chat, creative writing, and prompt rewriting for image generation. We also show a pilot study applying Gacha Decoding on protein design (Section 3.4). Complete details for our evaluation domains, baselines, and experimental setup are presented in Appendix C. Each domain consists of a set of prompts $\{x_1, \ldots, x_m\}$. To benchmark the performance of a given inference-time diverse generation method $\mathcal{M}$ with an LM θ, we generate N responses for each prompt x_i:

$$\mathcal{M}_\theta(x_i) \mapsto \left\{ y_i^{(1)}, \ldots, y_i^{(N)} \right\}. \tag{4}$$

We measure the quality of each response, and report a *quality-gated* diversity score for the full response set using domain-informed metrics. Specifically, for each prompt x_i, we only measure diversity among responses $\{y_i^{(j)}\}$ whose quality exceeds 95% of the average quality achieved via vanilla direct sampling on that prompt. This approach lets us normalize by quality on a prompt- and model-specific basis and prevent low-quality samples from inflating the diversity score (Shypula et al., 2025). Finally, we average metrics across all prompts of the domain. We also vary N to study how the diversity achieved by each method scales with the number of generations.

3.1.1 Models, baselines, and prior approaches

We compare Gacha Decoding with simple baselines and prior inference-time methods for diversifying model generations. For baselines, we consider (1) vanilla repeated **direct sampling** at temperature 1.0, (2) **high-temperature direct sampling** at temperature 1.5, and (3) **list prompting** to generate k (a hyperparameter) distinct responses at once. For prior approaches, we compare against (1) **Verbalized Sampling** (Zhang et al., 2025a), (2) **Direct Planning** (or Intent-Factored Generation (Ahmed et al., 2025)), (3) **Persona Prompting** (Ge et al., 2024; Santurkar et al., 2023), and (4) **Seed Prompting** (Misaki & Akiba, 2026). See Appendix C.1 for full details, including prompts.

Evaluated models. We evaluate on both open-weight (GPT-OSS, Qwen 3.5, and Inkling families) and closed source (Claude Opus 4.8 and GPT 5.5) language models. We always enable thinking mode and use each model's recommended sampling parameters. Note that the API models do not allow altering sampling temperature with thinking mode; we thus omit the methods which require higher temperature for those models. Please see Appendix C.2 for full inference details.

3.1.2 Open-ended evaluation domains

We describe our evaluation domains and prompts, and how the quality and diversity of the resulting responses are measured. See Appendix C.3 for complete details.

In-the-wild chat. Open-ended tasks such as advice-seeking, writing, and ideation are among the most common uses of chat systems (Chatterji et al., 2025). We evaluate on such queries through InfinityChat (Jiang et al., 2026). Following the original work, we measure diversity as response

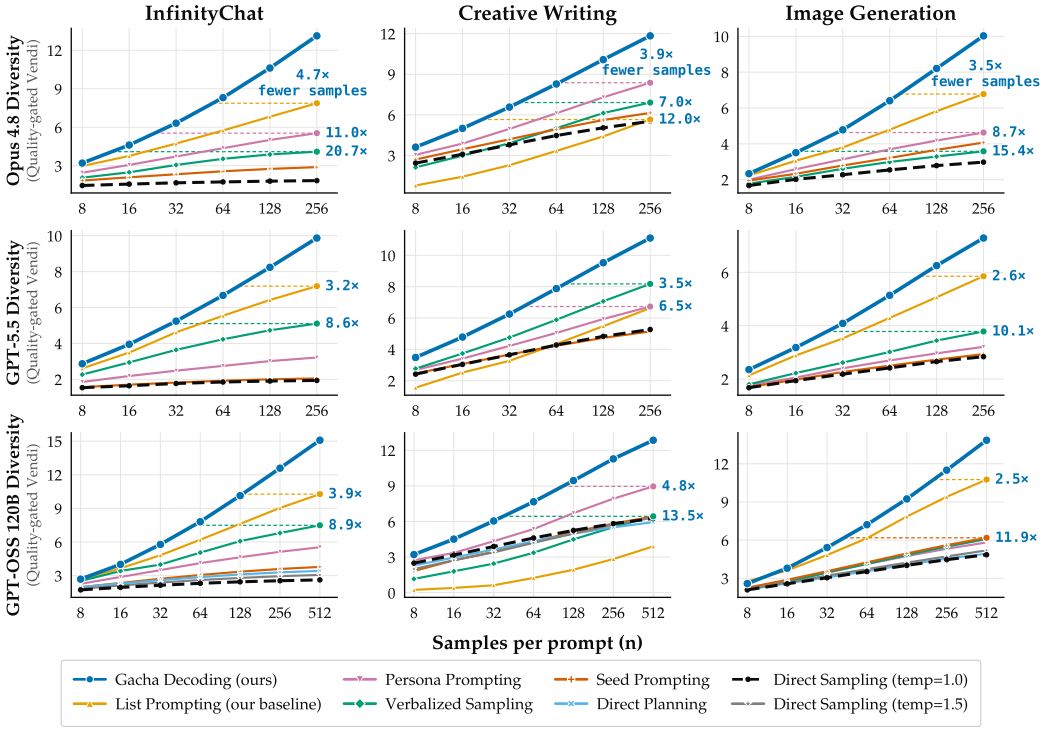


Figure 3: Gacha Decoding achieves much higher quality-controlled diversity than prior approaches across all models and task domains, with gains increasing at larger generation scales. This gain in Vendi translates to a stark improvement in mode-coverage sample efficiency (dashed lines). For example, on open-ended chat with Opus 4.8, Gacha Decoding requires 11.0x fewer samples to cover the same number of modes as 256 samples from persona prompting, the next-best prior approach.

embedding distance, reporting Vendi score (Friedman & Dieng, 2022). We measure quality as LM-judged response validity.

Creative writing. We construct an in-the-wild creative writing benchmark by filtering WildChat (Zhao et al., 2024) for fiction prompts. Following Zhang et al. (2025a), we score story quality from 0–20 using the LM-judge rubric of Paech (2023). We measure diversity following Springer et al. (2026); Kuhn et al. (2023): we (1) annotate stories along 8 semantic attributes (PREMISE, GENRE, SETTING, MAIN CHARACTER, NARRATOR POV, TONE, IMAGERY, RESOLUTION), (2) compute an embedding Vendi score for each, and (3) report their macro average. This decomposition captures high-level creative variation and reduces sensitivity to low-level lexical differences.

Image generation. Production text-to-image systems often use an LM to expand an initial user prompt x into a detailed caption x' before image generation: $x \mapsto x' \mapsto$ final image I. We evaluate applying diverse generation methods at this expansion stage to increase downstream image diversity. We sample open-ended prompts x from OneIGBench (Chang et al., 2026) and compare methods for generating diverse expanded captions x', rendering images with Qwen-Image-2512 (Wu et al., 2025b). We measure quality using OneIGBench's LM-judge rubrics for alignment with the original prompt x, and diversity using Vendi score over DreamSim embeddings (Fu et al., 2023).

3.2 Main Results

We show our main results in Fig. 3 and highlight examples of our method's generations in Fig. 1 and Appendix A.1. We provide extended results (including raw quality scores) in Appendix A.2.

Gacha Decoding strongly outperforms existing approaches, with gains continuing to increase at larger generation scales. Our marquee result is that Gacha Decoding yields significantly more diversity than prior approaches, at matched quality. For example, on Opus 4.8 with 256 generations per prompt, Gacha Decoding achieves a 7.1x increase in quality-gated diversity over direct sampling on open-ended chat. To put this in a sample efficiency perspective, our method requires 4.7x fewer

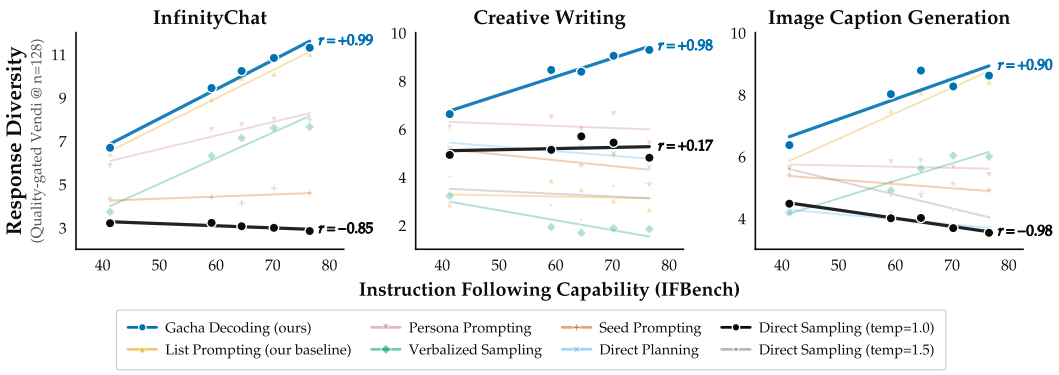


Figure 4: Our method derives diversity from an LM's instruction following ability as opposed to token entropy. Hence, our method is the only approach to consistently improve as the underlying LM's capabilities do, even when token entropy and diversity under entropy-reliant methods decline.

generations than our list prompting baseline—the next-best method—to reach the same level of effective mode coverage, matching 256 list generations with only $\sim$55 samples. Gacha Decoding similarly achieves 11.0x gain over persona prompting, the next-best prior approach, and 20.7x and near-38x efficiency gains over Verbalized Sampling and Seed Prompting, respectively.

Gacha Decoding can discover novel modes. These quantitative gains often reflect the emergence of qualitatively distinct response modes that other methods never reach. For example, when prompted to "*Write a horror story about a Singapore apartment*," Claude with Gacha Decoding generates several second-person stories, while all other methods remain limited to third-person narration (Fig. 6). On image generation, the diversity gains are often very visually apparent (Fig. 1).

List prompting is a strong baseline. Intriguingly, we find that simply prompting for a list of responses often outperforms all existing approaches we tried, in contrast with prior results (Zhang et al., 2025a). We conjecture that its stronger performance in our experiments stems from (1) carefully tuning the number of generations per list as a hyperparameter, and (2) evaluating at much larger generation budgets (up to 512 responses per prompt, versus around 8–32 in prior work).

3.3 Gacha scales with model IF capability and works under greedy decoding

Gacha Decoding is designed to derive diversity from an LM's instruction-following (IF) capability rather than its token entropy. This design suggests two predictions: (1) Gacha Decoding's diversity should increase with IF capability, and (2) Gacha Decoding should remain diverse even in an exterme case where the underlying LM has zero token entropy, such as under greedy decoding.

Scaling with instruction following. Prior work has often observed an inverse relationship between response diversity and model capability, as stronger models can have lower-entropy token distributions. In contrast, because Gacha Decoding relies on instruction following, its diversity should *improve* as the underlying LM improves. We test this across Qwen 3.5 models of varying parameter sizes (2B, 4B, 9B, 35B-A3B, 27B), plotting each method's quality-gated diversity against the model's IFBench instruction following score (Pyatkin et al., 2026) in Fig. 4.

Overall, we find that Gacha Decoding scales positively with IF capability (in contrast with direct sampling diversity, which indeed often decreases significantly), and is the only method to do so consistently across all domains. Some baselines, including list and persona prompting, show mixed scaling. We view these methods as partial realizations of our IF-centric view on diversity: list prompting, for example, can utilize IF to diversify responses *within* a single list, but diversity across independent lists still depends on token entropy. This works well for short-form domains like chat, where many responses can be considered within a single list, but breaks down for longer-form tasks such as creative writing. Gacha Decoding avoids this bottleneck via idea-space autoregression with external RNG, allowing IF-driven diversity to scale beyond what can be enumerated in a single call.

Diversity under greedy decoding. Even under greedy decoding (zero token entropy) on Qwen3.5 27B, GPT-OSS, and Inkling, we find Gacha retains its strong diversity, often outperforming all baselines run *with* sampling. In contrast, list-based methods degrade sharply without entropy (Table 3).

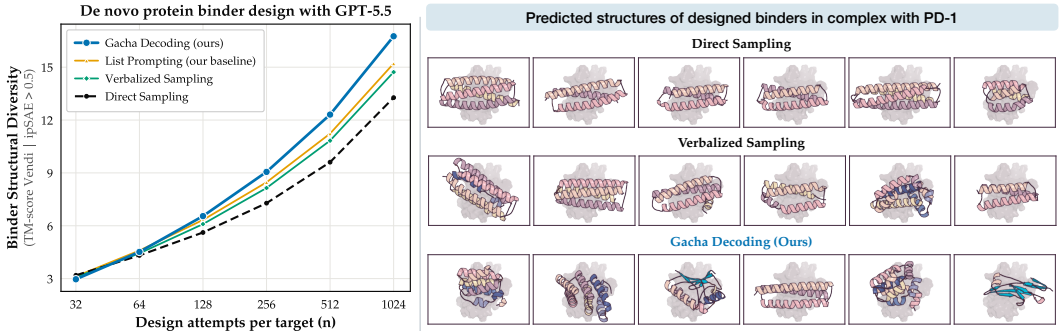


Figure 5: We use GPT 5.5 with Gacha Decoding to design diverse protein binders against downstream targets. Gacha Decoding yields the most structurally-diverse binders (TM-score Vendi) among designs passing a binding-quality threshold ($\text{ipSAE}_{\min} > 0.5$), with the gap widening with design budget. Against PD-1, Gacha Decoding discovers binders with mixed α/β secondary structure (β-sheets colored cyan), while Verbalized Sampling remains limited to all-helical folds.

3.4 A CASE STUDY IN PROTEIN DESIGN

As a pilot study, we apply Gacha Decoding to a downstream scientific workflow that directly benefits from increased diversity: de novo protein binder design. Generative protein models are increasingly used to accelerate drug discovery by designing candidate binders against therapeutic targets (Cao et al., 2022; Watson et al., 2023), but their outputs often concentrate on a narrow set of structural folds (Listov et al., 2024), reducing the effective number of designs explored. Recent text-to-protein models condition on natural language protein descriptions, allowing us to improve structure diversity via caption diversity analogously to image generation. Using GPT 5.5, we generate binder descriptions with Gacha Decoding and baseline methods for eight benchmark targets (e.g. PD-1, an immune checkpoint receptor targeted in cancer immunotherapy) (Pacesa et al., 2025), and realize each description as a target–binder complex with a fixed protein-generation model (Lee et al., 2026). We refold each design with its target using ESMFold2 (Candido et al., 2026), score predicted interface confidence using $\text{ipSAE}_{\min}$ (Dunbrack Jr, 2025), and retain designs with scores above 0.5 (Shanehsazzadeh, 2026). Among successful designs, we quantify structural diversity using Vendi score with pairwise Foldseek TM-scores as the similarity kernel (Van Kempen et al., 2024; Zhang & Skolnick, 2004). Please see Appendix D for full setup details. Results in Fig. 5 show that Gacha Decoding yields more diverse binder structures both quantitatively and qualitatively; for example, against PD-1, Gacha produces binders with mixed α/β secondary structure, whereas verbalized and list prompting remain restricted to all-helical folds.

4 ANALYSIS: WHY DOES GACHA DECODING WORK?

We ablate our three design choices from Section 2 to show their necessity in our method's success. Recall that Gacha autoregressively samples idea-level semantic decisions, using external RNG to randomize over LM-enumerated candidates at each local step. We ablate (details in Appendix A.4):

- **(A)** Randomizing aggressively over LM-curated ideas rather than tokens. We compare Gacha to extreme-temperature token sampling, using top-p (Holtzman et al., 2019) and top-K decoding (Fan et al., 2018) to randomize over an analogous "curated set" of top tokens at each step.
- **(B)** Sampling ideas by randomizing over LM-enumerated options. We consider no-RNG and no-enumeration Gacha variants. For each local decision, we instead directly prompt the model to "make a random choice" from its own listed options without external randomness (no-RNG), or sample an idea directly from the model via prompting (no-enumeration).
- **(C)** Sampling ideas autoregressively. We compare against two non-autoregressive Gacha variants: (1) *idea-list*, which enumerates k complete ideas in a single context (i.e., samples from the joint distribution over ideas) and (2) independent axes, which samples each local decision independently, treating the joint distribution as a product of marginals rather than conditionals.

We re-run our main experiments with these ablations on GPT-OSS-120B and Opus 4.8, finding that all variants are substantially weaker than Gacha Decoding (Table 1)—all our design choices are critical. See Appendix A.4 for corroborating results on Inkling and GPT 5.5, plus further discussion.

Method	InfinityChat	Creative Writing	Image Gen.
(A) Aggressively randomize over ideas vs tokens			
Gacha Decoding	**10.14**	**9.44**	**9.23**
Standard token sampling (T=1)	2.41	5.25	4.01
Token sampling (T=3, K=10)	3.28	3.07	4.66
Token sampling (T=3, K=20)	3.46	0.44	4.93
Token sampling (T=5, K=10)	3.19	0.31	4.90
Token sampling (T=5, K=20)	2.15	0.03	4.99
(B) Sample by decoupling enumeration + selection			
Gacha Decoding	**10.20**	**10.08**	**8.21**
No external RNG (LM selects choices from its list)	5.27	6.75	5.46
No enumeration (directly prompt LM for decision)	5.63	7.50	5.50
(C) Sample ideas autoregressively with local decisions			
Gacha Decoding	**10.20**	**10.08**	**8.21**
Idea-list (prompt LM to list k complete plans)	7.53	8.20	7.54
Independent axes (no conditioning on past decisions)	7.79	8.40	8.09

Table 1: We compare Gacha Decoding against ablations of our three design choices (Section 2), measuring diversity as quality-gated Vendi score (↑, n=128). We ablate **(A)** idea-space sampling via extreme-temperature top-K token sampling, **(B)** external randomization over LM-enumerated candidates via no-RNG and no-enumeration variants, and **(C)** sampling ideas autoregressively as a series of conditional local decisions by instead generating complete ideas directly (idea-list) or sampling decisions independently (independent axes). All ablations drop diversity—often sharply. Results on (A) are from GPT-OSS-120B, where temperature is configurable; (B, C) show Opus 4.8.

5 Related Work

(Lack of) language model diversity. A growing literature documents the limited diversity of LM generations on open-ended prompts (Zhang et al., 2025c; West & Potts, 2025; Kirk et al., 2023; Wu et al., 2025c; Xu et al., 2025), both within individual models and across model families (Jiang et al., 2026). This shortfall reduces user utility (Zhang et al., 2025c), homogenizes AI-assisted writing and decision-making (Abdulhai et al., 2026; Matz et al., 2025), and can constrain model development (Jung et al., 2026; Cui et al., 2025). Improving diversity has been challenging because scaling and capability post-training tend to make models *less* diverse (Karouzos et al., 2026; West & Potts, 2025), as training often reduces entropy. We show that this tension is not fundamental, leveraging the same IF capabilities that improve with post-training to drive greater diversity.

Approaches for improving diversity. Prior work has explored both training- and inference-time interventions. Training-time methods typically aim to make the LM's token distribution itself more diverse, for instance through diverse training data or new post-training objectives (Su et al., 2020; Zhang et al., 2024; Springer et al., 2026; Shi et al., 2018; Li et al., 2025a; Sorensen et al., 2026; Li et al., 2025b). Inference-time methods have traditionally reshaped the sampling distribution (Holtzman et al., 2019; Roemmele & Gordon, 2018), and more recently have steered generation toward different response modes by conditioning on seeds, personas, mode names, and other signals (Misaki & Akiba, 2026; Ge et al., 2024; Santurkar et al., 2023; Wu et al., 2025a). Our work crystallizes this latter line into an explicit instruction-following view of diversity, showing that external randomness combined with IF yields strong diversity without requiring LM token entropy.

6 Conclusion

In this work, we have shown that generation diversity can be treated as an instruction-following problem, shifting the locus of diversity away from LM token entropy—which can degrade with scale and post-training—toward a capability that improves as models do. Models need not encode all variation internally, but can instead act as reasoning components that convert external randomness from a surrounding system into useful, diverse responses. To do so, Gacha Decoding relies on models being able to enumerate distinct options when instructed; we find that modern LMs already do well enough, allowing us to strongly outperform prior methods with just a simple harness. Natural next steps include improving the harness by optimizing models end-to-end, and distilling the resulting diverse generations back into the LM. Overall, we view IF-centric diversity not as a replacement for methods that improve the model's intrinsic distribution, but as a complementary paradigm.

7 AI DISCLOSURE

In this work, we used generative AI tools to (1) get feedback on and polish our writing, (2) search for related work, and (3) assist with research execution, including writing code, running and giving feedback on experiments, and creating plots for our paper. We specifically use Claude Code (Opus 4.6+) and Codex (GPT 5.5+). We did not use AI for directly drafting paper sections or for proposing new methods, and our paper involves no synthetic datasets or mathematical proofs. We have reviewed all AI-generated work. For example, we manually reviewed the code for our experiments to ensure that all methods are implemented correctly, and that all baselines are compared fairly. We have also checked that reported numbers in AI-assisted figures match those in our raw results. We take responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of AI.

8 REPRODUCIBILITY

Our experiments use publicly-available evaluation datasets and models. All prompts and sampling configurations for reproducing our method and baselines are presented verbatim throughout the Appendix. For example, prompts for our method are in Appendix B; prompts for baselines are in Appendix C.1, and prompts for LM-judges used in our evaluations are in Appendix C.3.

REFERENCES

Marwa Abdulhai, Isadora White, Yanming Wan, Ibrahim Qureshi, Joel Leibo, Max Kleiman-Weiner, and Natasha Jaques. How llms distort our written language. *arXiv preprint arXiv:2603.18161*, 2026.

Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K Arora, Yu Bai, Bowen Baker, Haiming Bao, et al. gpt-oss-120b & gpt-oss-20b model card. *arXiv preprint arXiv:2508.10925*, 2025.

Eltayeb Ahmed, Uljad Berdica, Martha Elliott, Danijela Horak, and Jakob N Foerster. Intent factored generation: Unleashing the diversity in your language model. *arXiv preprint arXiv:2506.09659*, 2025.

Anthropic. Introducing claude opus 4.8. `https://www.anthropic.com/news/claude-opus-4-8`, May 2026. System card: `https://www.anthropic.com/claude-opus-4-8-system-card`. Model version `claude-opus-4-8`.

Salvatore Candido, Thomas Hayes, Alexander Derry, Roshan Rao, Zeming Lin, Robert Verkuil, Bryan Z Wu, Jin Sub Lee, Elise S Bruguera, Jehan A Keval, et al. Language modeling materializes a world model of protein biology. *bioRxiv*, pp. 2026–06, 2026.

Longxing Cao, Brian Coventry, Inna Goreshnik, Buwei Huang, William Sheffler, Joon Sung Park, Kevin M Jude, Iva Marković, Rameshwar U Kadam, Koen HG Verschueren, et al. Design of protein-binding proteins from the target structure alone. *Nature*, 605(7910):551–560, 2022.

Jingjing Chang, Yixiao Fang, Peng Xing, Shuhan Wu, Wei Cheng, Rui Wang, Xianfang Zeng, Gang Yu, and Hai-Bao Chen. Oneig-bench: Omni-dimensional nuanced evaluation for image generation. *Advances in Neural Information Processing Systems*, 38, 2026.

Aaron Chatterji, Thomas Cunningham, David J Deming, Zoe Hitzig, Christopher Ong, Carl Yan Shan, and Kevin Wadman. How people use chatgpt. Technical report, National Bureau of Economic Research, 2025.

Ganqu Cui, Yuchen Zhang, Jiacheng Chen, Lifan Yuan, Zhi Wang, Yuxin Zuo, Haozhan Li, Yuchen Fan, Huayu Chen, Weize Chen, et al. The entropy mechanism of reinforcement learning for reasoning language models. *arXiv preprint arXiv:2505.22617*, 2025.

Roland L Dunbrack Jr. Rēs ipsae loquunt: What's wrong with alphafold's iptm score and how to fix it. *Biorxiv*, 2025.

Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 889–898, 2018.

Dan Friedman and Adji Bousso Dieng. The vendi score: A diversity evaluation metric for machine learning. *arXiv preprint arXiv:2210.02410*, 2022.

Stephanie Fu, Netanel Tamir, Shobhita Sundaram, Lucy Chai, Richard Zhang, Tali Dekel, and Phillip Isola. Dreamsim: Learning new dimensions of human visual similarity using synthetic data. *arXiv preprint arXiv:2306.09344*, 2023.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling synthetic data creation with 1,000,000,000 personas. *arXiv preprint arXiv:2406.20094*, 2024.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, and Yejin Choi. Artificial hivemind: The open-ended homogeneity of language models (and beyond). *Advances in Neural Information Processing Systems*, 38, 2026.

Jaehun Jung, Seungju Han, Ximing Lu, Skyler Hallinan, David Acuna, Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, and Yejin Choi. Prismatic synthesis: Gradient-based data diversification boosts generalization in llm reasoning. *Advances in Neural Information Processing Systems*, 38:90649–90685, 2026.

Constantinos Karouzos, Xingwei Tan, and Nikolaos Aletras. Where does output diversity collapse in post-training? *arXiv preprint arXiv:2604.16027*, 2026.

Robert Kirk, Ishita Mediratta, Christoforos Nalmpantis, Jelena Luketina, Eric Hambro, Edward Grefenstette, and Roberta Raileanu. Understanding the effects of rlhf on llm generalisation and diversity. *arXiv preprint arXiv:2310.06452*, 2023.

Lorenz Kuhn, Yarin Gal, and Sebastian Farquhar. Semantic uncertainty: Linguistic invariances for uncertainty estimation in natural language generation. *arXiv preprint arXiv:2302.09664*, 2023.

Joseph Lee et al. Prompteina, 2026. Forward cite for a forthcoming model and paper release.

Tianjian Li, Yiming Zhang, Ping Yu, Swarnadeep Saha, Daniel Khashabi, Jason Weston, Jack Lanchantin, and Tianlu Wang. Jointly reinforcing diversity and quality in language model generations. *arXiv preprint arXiv:2509.02534*, 2025a.

Ziniu Li, Congliang Chen, Tian Xu, Zeyu Qin, Jiancong Xiao, Zhi-Quan Luo, and Ruoyu Sun. Preserving diversity in supervised fine-tuning of large language models. In *International Conference on Learning Representations*, volume 2025, pp. 66127–66154, 2025b.

Dina Listov, Casper A Goverde, Bruno E Correia, and Sarel Jacob Fleishman. Opportunities and challenges in design and optimization of protein function. *Nature Reviews Molecular Cell Biology*, 25(8):639–653, 2024.

Sandra C Matz, C Blaine Horton, and Sofie Goethals. The basic b*** effect: The use of llm-based agents reduces the distinctiveness and diversity of people's choices. *arXiv preprint arXiv:2509.02910*, 2025.

Yev Meyer and Dane Corneil. Nemotron-Personas-USA: Synthetic personas aligned to real-world distributions, June 2025. URL `https://huggingface.co/datasets/nvidia/Nemotron-Personas-USA`.

Kou Misaki and Takuya Akiba. String seed of thought: Prompting llms for distribution-faithful and diverse generation. In *International Conference on Learning Representations*, volume 2026, pp. 89078–89136, 2026.

OpenAI. Introducing gpt-5.5. `https://openai.com/index/introducing-gpt-5-5/`, April 2026. Model version `gpt-5.5`.

Martin Pacesa, Lennart Nickel, Christian Schellhaas, Joseph Schmidt, Ekaterina Pyatova, Lucas Kissling, Patrick Barendse, Jagrity Choudhury, Srajan Kapoor, Ana Alcaraz-Serna, et al. One-shot design of functional protein binders with bindcraft. *Nature*, 646(8084):483–492, 2025.

Samuel J Paech. Eq-bench: An emotional intelligence benchmark for large language models. *arXiv preprint arXiv:2312.06281*, 2023.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hanna Hajishirzi. Generalizing verifiable instruction following. *Advances in Neural Information Processing Systems*, 38, 2026.

Qwen Team. Qwen3.5: Towards native multimodal agents, February 2026. URL https://qwen.ai/blog?id=qwen3.5.

Melissa Roemmele and Andrew S Gordon. Automated assistance for creative writing with an rnn language model. In *Companion proceedings of the 23rd international conference on intelligent user interfaces*, pp. 1–2, 2018.

Shibani Santurkar, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto. Whose opinions do language models reflect? In *International conference on machine learning*, pp. 29971–30004. PMLR, 2023.

Amir Shanehsazzadeh. Autonomous de novo protein binder design with claude, 2026.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. Toward diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*, 2018.

Alexander Shypula, Shuo Li, Botong Zhang, Vishakh Padmakumar, Kayo Yin, and Osbert Bastani. Evaluating the diversity and quality of llm generated content. *arXiv preprint arXiv:2504.12522*, 2025.

Taylor Sorensen, Benjamin Newman, Jared Moore, Chan Young Park, Jillian Fisher, Niloofar Mireshghallah, Liwei Jiang, and Yejin Choi. Spectrum tuning: Post-training for distributional coverage and in-context steerability. In *International Conference on Learning Representations*, volume 2026, pp. 76138–76212, 2026.

Jacob Mitchell Springer, Madhu Advani, Lukas Aichberger, Arwen Bradley, Eran Malach, Omid Saremi, Sinead Williamson, Preetum Nakkiran, Etai Littwin, and Aditi Raghunathan. Annotations mitigate post-training mode collapse. *arXiv preprint arXiv:2605.09995*, 2026.

Hui Su, Xiaoyu Shen, Sanqiang Zhao, Zhou Xiao, Pengwei Hu, Randy Zhong, Cheng Niu, and Jie Zhou. Diversifying dialogue generation with non-conversational text. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7087–7097, 2020.

Gemma Team, Sherif El Abd, Vaibhav Aggarwal, Robin Algayres, Alek Andreev, Olivier Bachem, Ian Ballantyne, Cormac Brick, Victor Cărbune, Michelle Casbon, et al. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Thinking Machines Lab. Inkling: Our open-weights model. https://thinkingmachines.ai/news/introducing-inkling/, July 2026. Inkling-Small: https://thinkingmachines.ai/news/inkling-small/.

Michel Van Kempen, Stephanie S Kim, Charlotte Tumescheit, Milot Mirdita, Jeongjae Lee, Cameron LM Gilchrist, Johannes Söding, and Martin Steinegger. Fast and accurate protein structure search with foldseek. *Nature biotechnology*, 42(2):243–246, 2024.

Joseph L Watson, David Juergens, Nathaniel R Bennett, Brian L Trippe, Jason Yim, Helen E Eisenach, Woody Ahern, Andrew J Borst, Robert J Ragotte, Lukas F Milles, et al. De novo design of protein structure and function with rfdiffusion. *Nature*, 620(7976):1089–1100, 2023.

Peter West and Christopher Potts. Base models beat aligned models at randomness and creativity. *arXiv preprint arXiv:2505.00047*, 2025.

Chen Henry Wu, Sachin Goyal, and Aditi Raghunathan. Mode-conditioning unlocks superior test-time scaling. *arXiv preprint arXiv:2512.01127*, 2025a.

Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, Yuxiang Chen, Zecheng Tang, Zekai Zhang, Zhengyi Wang, An Yang, Bowen Yu, Chen Cheng, Dayiheng Liu, Deqing Li, Hang Zhang, Hao Meng, Hu Wei, Jingyuan Ni, Kai Chen, Kuan Cao, Liang Peng, Lin Qu, Minggang Wu, Peng Wang, Shuting Yu, Tingkun Wen, Wensen Feng, Xiaoxiao Xu, Yi Wang, Yichang Zhang, Yongqiang Zhu, Yujia Wu, Yuxuan Cai, and Zenan Liu. Qwen-image technical report, 2025b. URL https://arxiv.org/abs/2508.02324.

Fan Wu, Emily Black, and Varun Chandrasekaran. Generative monoculture in large language models. In *International Conference on Learning Representations*, volume 2025, pp. 33068–33107, 2025c.

Weijia Xu, Nebojsa Jojic, Sudha Rao, Chris Brockett, and Bill Dolan. Echoes in ai: Quantifying lack of plot diversity in llm outputs. *Proceedings of the National Academy of Sciences*, 122(35): e2504966122, 2025.

Hiromu Yakura, Ezequiel Lopez-Lopez, Levin Brinkmann, Ignacio de la Serna, Lara Kirfel, Prateek Gupta, Ivan Soraperra, Thomas F Eisenmann, Dirk U Wulff, and Iyad Rahwan. Empirical evidence of large language model's influence on human spoken communication. *arXiv preprint arXiv:2409.01754*, 2024.

Lili Yao, Nanyun Peng, Ralph Weischedel, Kevin Knight, Dongyan Zhao, and Rui Yan. Plan-and-write: Towards better automatic storytelling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pp. 7378–7385, 2019.

Cheng Zhang, Rui Xin, and Chudi Zhong. Where you inject diversity matters: A unified framework for diverse generation. *arXiv preprint arXiv:2606.10302*, 2026.

Jiayi Zhang, Simon Yu, Derek Chong, Anthony Sicilia, Michael R Tomz, Christopher D Manning, and Weiyan Shi. Verbalized sampling: How to mitigate mode collapse and unlock llm diversity. *arXiv preprint arXiv:2510.01171*, 2025a.

Yang Zhang and Jeffrey Skolnick. Scoring function for automated assessment of protein structure template quality. *Proteins: Structure, Function, and Bioinformatics*, 57(4):702–710, 2004.

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

Yiming Zhang, Avi Schwarzschild, Nicholas Carlini, Zico Kolter, and Daphne Ippolito. Forcing diffuse distributions out of language models. *arXiv preprint arXiv:2404.10859*, 2024.

Yiming Zhang, Harshita Diddee, Susan Holm, Hanchen Liu, Xinyue Liu, Vinay Samuel, Barry Wang, and Daphne Ippolito. Noveltybench: Evaluating language models for humanlike diversity. *arXiv preprint arXiv:2504.05228*, 2025c.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody H Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems*, 37:62557–62583, 2024.

A Extended results

A.1 Qualitative examples

See Fig. 6 for qualitative story examples generated by Gacha Decoding and baselines.

Direct sampling — third person

The flyer in the elevator promised everything **Mei** wanted: a two-bedroom HDB flat in Tampines, resale price forty thousand below market. **She** should have known better. The auntie who sold it barely made eye contact. "Renovate quick," she said, pressing the keys into **Mei's** palm. "Don't sleep in the small room first." **Mei** laughed it off. Superstition.

Verbalized sampling — third person

Ravi bought the resale flat because it was cheap—suspiciously cheap for a five-room unit in Sengkang. The agent had smiled too much, talked too fast, and never once mentioned why the previous family had left after only two months. The first strange thing was the humming. It came from the master bedroom wardrobe, a low tuneless drone that **Ravi** first mistook for a fault in the aircon.

Gacha Decoding — second person

You move into Block 114 because it is cheap, and it is cheap because it is dying. The agent doesn't say this. She says *quiet*, she says *character*, she says *the en-bloc talks fell through so prices are soft*. But **you** count the dark windows as **you** crane **your** neck upward in the void deck, and **you** understand. Of the fifty-odd units stacked above **you**, perhaps six are lit.

Figure 6: Gacha Decoding can discover novel modes, surfacing several stories with a second-person point-of-view when no other approaches do so after 256 samples. The original task prompt is "Write a horror story about a Singapore apartment," drawn from WildChat (Zhao et al., 2024) and included amongst our story evaluation prompts.

A.2 Additional main results

See Fig. 7 for corroborating main results on Inkling Large, which we defer here for the sake of space. See Table 2 for the concrete quality and diversity metrics numbers from our main experiments.

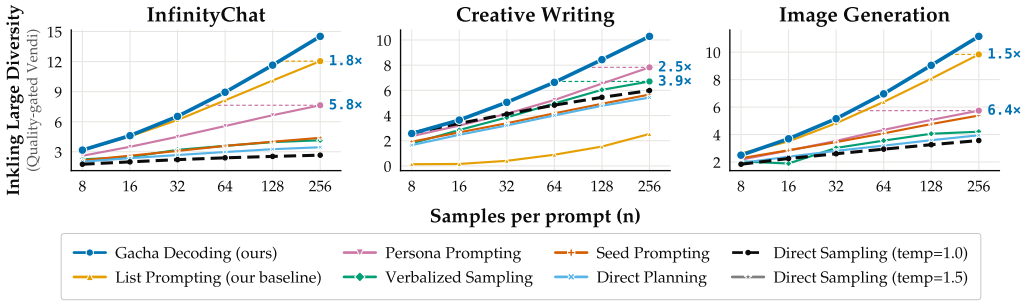


Figure 7: Results on Inkling corroborate our main results in Fig. 3; Gacha Decoding outperforms all existing baselines.

A.3 Diversity under greedy decoding

See Table 3 for results comparing Gacha Decoding and baselines under standard token sampling (temperature=1.0) and greedy decoding. Overall, Gacha Decoding retains its diversity while list-based methods drop severely. Persona and seed prompting also retain much of their diversity because they also externalize some variation from the LM. Gacha Decoding is the only method that combines strong diversity with independence from token entropy.

Method	InfinityChat			Creative Writing			Image Generation		
	Quality-gated diversity (raw div.)	Avg. quality of passing (raw qual.)	Quality-gate pass rate (%)	Quality-gated diversity (raw div.)	Avg. quality of passing (raw qual.)	Quality-gate pass rate (%)	Quality-gated diversity (raw div.)	Avg. quality of passing (raw qual.)	Quality-gate pass rate (%)
Claude Opus 4.8									
Gacha (ours)	**13.12** (15.14)	1.00 (0.80)	79.5	**11.84** (11.92)	17.4 (17.3)	97.8	**10.03** (13.37)	0.94 (0.82)	57.6
List	7.87 (8.62)	1.00 (0.89)	88.7	5.65 (9.67)	14.9 (11.8)	19.0	6.78 (7.89)	0.95 (0.85)	66.2
Verbalized	4.10 (4.19)	1.00 (0.95)	95.4	6.91 (8.31)	15.3 (14.1)	46.8	3.58 (3.92)	0.96 (0.87)	67.6
Persona	5.54 (5.67)	1.00 (0.94)	93.8	8.37 (8.84)	16.4 (15.7)	83.8	4.63 (5.04)	0.95 (0.87)	74.9
Seed	2.89 (2.90)	1.00 (0.98)	97.7	6.14 (6.16)	16.6 (16.5)	97.0	4.07 (4.43)	0.95 (0.86)	73.1
Direct (T=1)	1.84 (1.85)	1.00 (0.95)	94.8	5.53 (5.62)	15.9 (15.2)	79.5	2.98 (3.11)	0.94 (0.88)	76.4
GPT-5.5									
Gacha (ours)	**9.87** (10.61)	1.00 (0.90)	89.6	**11.10** (11.13)	17.5 (17.5)	98.9	**7.29** (8.36)	0.95 (0.85)	68.7
List	7.19 (7.50)	1.00 (0.91)	91.3	6.63 (8.70)	15.2 (13.7)	32.4	5.86 (6.56)	0.94 (0.86)	69.9
Verbalized	5.11 (5.15)	1.00 (0.96)	96.0	8.18 (8.71)	16.0 (15.6)	73.9	3.78 (4.04)	0.95 (0.87)	72.9
Persona	3.22 (3.22)	1.00 (0.98)	98.2	6.73 (7.09)	16.0 (15.6)	80.6	3.21 (3.35)	0.95 (0.88)	77.2
Seed	2.05 (2.06)	1.00 (0.98)	98.2	5.14 (5.21)	16.0 (15.7)	87.9	2.94 (3.04)	0.95 (0.88)	77.1
Direct (T=1)	1.94 (1.94)	1.00 (0.98)	97.5	5.27 (5.37)	16.0 (15.5)	82.6	2.84 (2.94)	0.95 (0.88)	77.6
GPT-OSS 120B									
Gacha (ours)	**15.08** (19.20)	1.00 (0.63)	63.3	**12.84** (13.73)	14.1 (13.4)	77.1	**13.85** (17.09)	0.92 (0.82)	61.0
List	10.27 (12.57)	1.00 (0.76)	76.0	3.90 (10.86)	11.4 (6.8)	6.1	10.76 (12.40)	0.93 (0.84)	66.9
Verbalized	7.49 (8.61)	1.00 (0.80)	79.8	6.43 (9.11)	12.8 (10.6)	25.4	6.10 (6.65)	0.93 (0.85)	73.1
Persona	5.53 (5.70)	1.00 (0.89)	89.1	8.95 (10.02)	13.7 (12.1)	57.2	5.81 (6.11)	0.93 (0.87)	77.6
Seed	3.77 (3.98)	1.00 (0.87)	87.3	6.42 (7.11)	13.4 (11.6)	47.3	6.18 (6.55)	0.93 (0.86)	75.5
Direct Planning	3.42 (3.47)	1.00 (0.93)	93.4	5.94 (6.27)	13.6 (12.5)	65.0	4.93 (5.11)	0.93 (0.87)	78.6
Direct (T=1.5)	3.06 (3.11)	1.00 (0.89)	89.5	6.32 (7.06)	13.6 (11.5)	52.1	5.19 (5.43)	0.93 (0.87)	79.2
Direct (T=1)	2.62 (2.64)	1.00 (0.92)	92.4	6.26 (6.45)	13.8 (12.8)	74.5	4.85 (5.04)	0.93 (0.87)	79.4
Inkling Large									
Gacha (ours)	**14.52** (17.65)	1.00 (0.73)	72.7	**10.29** (12.01)	17.0 (16.0)	51.9	**11.15** (14.28)	0.93 (0.83)	58.8
List	12.04 (14.41)	1.00 (0.77)	77.2	2.54 (9.98)	15.9 (11.7)	3.5	9.83 (12.12)	0.93 (0.84)	60.7
Verbalized	4.15 (4.45)	1.00 (0.89)	89.1	6.71 (8.24)	16.8 (15.8)	43.9	4.21 (4.54)	0.94 (0.87)	74.5
Persona	7.64 (8.16)	1.00 (0.85)	84.8	7.82 (9.56)	16.8 (15.8)	47.9	5.74 (6.25)	0.94 (0.87)	74.1
Seed	4.39 (4.65)	1.00 (0.87)	86.5	5.67 (6.58)	16.9 (15.6)	43.8	5.39 (5.80)	0.94 (0.87)	75.4
Direct Planning	3.45 (3.50)	1.00 (0.95)	95.2	5.43 (6.48)	16.8 (14.5)	39.8	3.95 (4.13)	0.94 (0.88)	78.0
Direct (T=1)	2.66 (2.67)	1.00 (0.98)	97.5	5.98 (6.08)	17.2 (17.0)	88.9	3.57 (3.71)	0.94 (0.88)	80.4

Table 2: Extended main results for our experiments in Section 3, showing the concrete quality and diversity scores underlying Figs. 3 and 7. **Quality-gated diversity**: Vendi score over the samples that pass a quality gate, our main metric. We show raw Vendi over all samples—including low-quality ones—in parentheses. **Average quality of passing samples**: mean judge quality of the passing samples (chat response validity, binary 0/1; story rubric-judged quality, 0–20; image rubric-judged alignment, 0–1), with the mean over all samples in parentheses. **Quality-gate pass rate**: the percentage of samples that pass the quality gate, defined to be $0.95\times$ the model's direct-sampling mean on the same prompt. We show results at the highest number of samples per prompt considered for each model (n=512 for GPT-OSS, and n=256 elsewhere).

Model	Method	InfinityChat			Creative Writing			Image Generation		
		T=1	Greedy	Δ	T=1	Greedy	Δ	T=1	Greedy	Δ
GPT-OSS 120B	**Gacha (ours)**	**10.08**	**9.62**	-4.6%	**9.44**	**9.20**	-2.5%	**9.23**	**8.82**	-4.5%
	List	7.62	4.71	-38.1%	1.93	0.06	-96.8%	7.84	4.48	-42.9%
	Verbalized	6.07	3.83	-36.9%	4.49	0.65	-85.5%	4.73	3.19	-32.7%
	Persona	4.64	4.64	+0.2%	6.71	6.58	-1.8%	4.74	4.67	-1.5%
	Seed	3.35	3.08	-8.0%	5.12	4.74	-7.3%	4.95	4.68	-5.5%
Inkling Large	**Gacha (ours)**	**11.65**	**11.65**	0.0%	**8.44**	**8.03**	-4.8%	**9.04**	**9.05**	+0.1%
	List	10.11	5.90	-41.7%	1.54	0.06	-96.0%	8.07	4.94	-38.8%
	Verbalized	3.96	1.69	-57.4%	6.04	0.54	-91.0%	4.06	1.23	-69.6%
	Persona	6.65	6.56	-1.4%	6.54	6.07	-7.1%	5.07	4.90	-3.2%
	Seed	4.00	3.85	-3.9%	4.93	4.45	-9.9%	4.76	4.64	-2.6%
Qwen3.5-27B	**Gacha (ours)**	**11.29**	**10.62**	-6.0%	**9.29**	**8.37**	-9.9%	**8.61**	**8.31**	-3.5%
	List	11.01	5.29	-51.9%	2.67	0.00	-100.0%	8.42	4.64	-44.8%
	Verbalized	7.64	4.31	-43.6%	1.84	0.00	-100.0%	5.99	3.57	-40.4%
	Persona	7.92	7.38	-6.8%	5.40	6.18	+14.4%	5.40	5.06	-6.3%
	Seed	4.60	3.54	-23.0%	3.73	4.00	+7.1%	4.91	4.07	-17.1%

Table 3: We show quality-gated Vendi score diversity (↑, results at $n = 128$) with Gacha Decoding and other inference-time methods under (a) standard token sampling (temperature=1.0) and (b) greedy decoding. Because Gacha Decoding externalizes randomness from the LM and does not rely on token entropy, going to greedy decoding only negligibly affects the diversity it elicits (Δ). The near-complete drop in Verbalized Sampling and list prompting diversity on creative writing is because these methods often degenerate into nonsensical responses under greedy decoding, such that few if any responses pass the quality gate.

Method	InfinityChat	Creative Writing	Image Gen.
(A) Aggressively randomize over ideas vs tokens			
Gacha Decoding	**11.65**	**8.44**	**9.04**
Standard token sampling (T=1)	2.54	5.45	3.26
Token sampling (T=3, K=10)	3.07	2.56	4.25
Token sampling (T=3, K=20)	2.16	0.00	4.90
Token sampling (T=5, K=10)	1.79	0.03	5.02
Token sampling (T=5, K=20)	0.42	0.00	5.29
(B) Decouple enumeration from selection			
Gacha Decoding	**8.24**	**9.53**	**6.25**
No enumeration (directly prompt LM for decision)	4.58	6.38	4.42
No external RNG (LM selects choices from its list)	3.95	6.41	5.08
(C) Sample ideas autoregressively			
Gacha Decoding	**8.24**	**9.53**	**6.25**
Idea-list (prompt LM to list k complete plans)	5.96	7.95	5.43
Independent axes (no conditioning on past decisions)	6.34	8.20	6.02

Table 4: Additional ablation results to corroborate Table 1. Panel A shows results with Inkling Large, panels B and C show GPT 5.5.

A.4 ADDITIONAL ABLATION RESULTS

We provide additional discussion and results for the main ablations shown in Table 1. Each ablation removes one of the three design choices underlying Gacha Decoding: (A) randomizing over semantic ideas rather than tokens, (B) decoupling enumeration from selection with external RNG, and (C) sampling ideas autoregressively. Our main ablations table showed results on GPT-OSS for (A), as it allows configuring sampling parameters, and results on Opus 4.8 for (B, C). Here, we show corroborating results on Inkling and GPT 5.5 (Table 4).

(A) Idea-space vs. token-space randomization. Gacha Decoding works in idea space, aggressively randomizing over curated sets of LM-generated ideas by using external RNG to uniformly sample. To test whether the same strategy can be applied at the token level, we compare against extreme-temperature token sampling with top-p (Holtzman et al., 2019) and top-K (Fan et al., 2018) decoding. At each decoding step, top-K leverages the LM to obtain an analogous "curated set" of most likely next tokens; increasing temperature makes sampling over this set progressively more uniform. We evaluate several values of K and temperature, fixing $p = 0.95$. We find that even for relatively small K, increasing temperature beyond 3—perhaps unsurprisingly—greatly damages the coherence of responses and consequently limits quality-filtered diversity. Note that temperature 3 is still far from uniform sampling, which corresponds to the infinite temperature limit. Working in idea space is critical for the aggressive randomness that enables Gacha Decoding.

(B) Decoupling enumeration from selection. Gacha Decoding relies on external RNG to uniformly sample ideas from LM-enumerated candidates, forming a sequence of randomized idea-level decisions. We show that this enumerate-select setup for idea sampling is critical. We compare Gacha Decoding against a (1) no-enumeration variant and a (2) no-RNG variant.

In the *no-enumeration* variant, instead of instructing the LM to enumerate a list of plausible choices for each decision, we directly prompt the model for a single choice conditioned on the task and previous decisions. This ablation preserves the autoregressive decision structure, but variation in each choice now comes from the model's token distribution. One can also view this ablation as an autoregressive extension of Intent-Factored Generation.

In the *no-RNG* variant, the LM still enumerates the same candidate options as in Gacha Decoding, but we ask the model itself to choose randomly among them rather than using external RNG. Thus, the support of the conditional idea distribution is unchanged, but selection is now delegated back to the LM itself. As we saw earlier in Table 1 and again in Table 4, removing external RNG sharply drops diversity. Intuitively, the model's own sampled choice tends to collapse towards the middle-rank options within the list and is far from uniform; LMs are poor at giving a uniform output token distribution over option indices. As an illustrative anecdote, we find even base models gravitate towards the number 37 when asked for a uniform random number from 1-100, mirroring a bias in humans[2].

(C) Autoregressive idea sampling. Gacha Decoding samples ideas autoregressively, where each decision is sampled conditioned on the previous ones. This allows earlier random choices to inform and diversify the options for later ones, enabling response diversity to come from the model's ability to enumerate ideas under different contexts as opposed to token entropy. We ablate Gacha Decoding's autoregressive structure by comparing against two non-autoregressive variants. First, we enumerate k complete ideas in a single context and sample one idea directly. This "*idea-list*" variant can be viewed as enumerating directly from the joint distribution over ideas, and relies on token entropy to diversify across idea lists. Second, we decompose the task into the same semantic decision axes as Gacha Decoding but sample each axis independently rather than sequentially. This "*independent-axes*" variant approximates the joint distribution as a product of marginals rather than a chain of conditionals, and does not enable one decision to diversify the options for others. Results in Tables 1 and 4 show that Gacha Decoding significantly outperforms both variants.

[2]See `https://www.youtube.com/watch?v=d6iQrh2TK98`, which runs an informal n=200,000 survey and finds humans gravitate towards 37.

B Details on Gacha Decoding

We present the prompt templates for each step in the Gacha harness, followed by concrete domain-dependent template values we use in our main experiments. {task} is always the user prompt x. Model replies are parsed as JSON. Each LM call is performed at the default sampling parameters of the model (typically temperature 1.0, with reasoning on, see Appendix C.2) as documented in the corresponding model cards.

Step 1: Plan the decision axes. To sample a response y to some prompt x, we begin by planning the decision axes. We keep the first T decision axes from the plan, where T is a hyperparameter (see below for concrete per-domain values). We did not extensively sweep T for each domain, and believe that optimizing this hyperparameter may lead to additional gains in future work.

Planning decision axes

```
Design a schema that partitions the space of possible outputs for the
task below into sequential decision points, so that distinct paths
through the schema yield {goal}. Use AT MOST {T} questions total. Each
decision point is phrased as a question, one per turn; its options are
enumerated later, conditioned on all earlier answers.

Rules:
- Only partition what the task leaves free; never re-decide what it
fixes.
- First brainstorm the set of questions, and then commit to the
ordering. A good ordering should place questions with the fewest
dependencies first, and leave the largest room for variation in later
questions.
- Prefer simple, enumerable questions at the start, deferring
finer-grained, abstract, or interpretive questions for later.
- Each question must introduce new high-level information that
meaningfully steers the output.
- Give each question a two-or-three-word name for labeling its answer.

Task: {task}
Output exactly: {"questions": [{"name": "...", "ask": "..."}]}
```

- **Story:** $T = 4$; {goal} = "stories that vary widely in: genre, setting, narrative tone, narrator and point of view, premise, emotional resolution, dominant imagery, main character". These attributes are chosen to align with the evaluation attributes (see Section 3) and to inform the LM of how the diversity of its stories will be judged. We include these attributes for all baseline prompting approaches as well to ensure fair comparison.
- **Chat:** $T = 4$; {goal} = "diverse high-quality responses".
- **Image:** $T = 6$; {goal} = "diverse, high-quality generated images that would appear perceptually distinct even at a glance". For image we use a condensed version of the planner that drops the ***Rules*** block, as we found a simpler prompt worked well in this setting; we view further per-domain prompt optimization as future work.

(Looped) Steps 2–4: Enumerate options for decision t, sample one choice, update the context for future decisions. Now that we have planned the decision axes, we make the decisions autoregressively within a multi-turn LM conversation (separate context from the earlier planning stage). To pick a choice z_t for the decision axis at time t, we prompt the model with

Decision turn t

```
Chosen {decision_name_(t-1)}: z_(t-1)

{decision_question_t} List at least {m} distinct options. Make sure
that each option is consistent with all previously made choices.
Output exactly:
{"options": ["...", "...", ...]}
```

as a new turn in a growing multi-turn conversation. For the first decision at time $t = 1$, since there is no previous decision, we prepend the original prompt x.

At each step, the model returns a candidate set of enumerated options $\mathcal{C}_t$ as a JSON, from which the harness draws a single option z_t uniformly with an RNG tool. This decision is then used to construct the prompt for the next turn deciding z_{t+1}. By adopting a multi-turn setup, we ensure that subsequent decisions are steered by all earlier decisions $z_{\leq t}$, enumerated options, and sampled outcomes. If the LM call errors on any single turn (e.g., if the returned JSON is unparseable), then that decision is simply skipped.

For **story** and **image**, we use lists of size $m = 32$. For **chat**, we use $m = 24$. Again, this is not thoroughly optimized; other choices of this hyperparameter may yield stronger results.

Step 5: Realize. One final menu turn is appended to the same multi-turn conversation, over complete ideas rather than a single axis, and sampled the same way ($k = 8$ in all domains). A fresh call then renders the prompt and all sampled choices into the response.

Idea turn (final turn within multi-turn conversation)

```
Chosen {decision_name_T}: {z_T}.

Given every choice above, list {k} distinct creative angles, themes, or
approaches for the task. Be specific and varied.

Task: {task}
Output exactly:
{"options": ["...", "...", ...]}
```

Executor (fresh context)

```
Your task: {task}

Planned decisions:
{decision_name_1}: {z_1}
...
{decision_name_T}: {z_T}
Overall approach (the guiding take | write the whole response as this):
{z_idea}

Write the single best {artifact} that satisfies the request while (a)
satisfying the task and all its constraints and (b) realizing every
design choice above. If a choice cannot be honored within the request's
exact constraints, sacrifice the choice, not the request. Give the
answer itself without preamble or meta-commentary.
```

Here, {artifact} is replaced with "story" / "response" / "caption" for the story / chat / image domains, respectively.

C Details on Experimental Setup

C.1 Baselines and Prior Approaches

We detail our baselines, including exact prompts used to implement them. In general, we place method-specific instructions into the system prompt (following the original works) and the user turn is simply the evaluation task prompt x. As with Gacha Decoding, unless otherwise noted, every LM call uses the model's default sampling parameters.

To first name our baselines in more detail, we compare Gacha Decoding against:

1. **Direct sampling.** We sample IID responses to the original prompt using each model's default temperature. We also consider **high-temperature direct sampling**, using temperature 1.5.

2. **List prompting.** We instruct the model to produce k (a hyperparameter) distinct and varied responses in a single generation. Since earlier responses remain in context while later responses are generated, this method allows the model to explicitly avoid repeating its previous outputs.
3. **Verbalized Sampling** (Zhang et al., 2025a). We ask the model to verbalize a distribution over possible responses, producing k responses together with their probabilities in a single generation.
4. **Direct Planning (Intent-Factored Generation)** (Ahmed et al., 2025). We generate in two stages; we first sample a high-level idea or plan from the LM at a higher-than-typical temperature (1.2), and then produce a final response conditioned on the idea in a separate call.
5. **Seed Prompting** (Misaki & Akiba, 2026). We prepend a random ASCII string (e.g. "`aYn67`...") to the task prompt and instruct the model to generate a response informed by that string.
6. **Persona Prompting** (Ge et al., 2024; Santurkar et al., 2023). We instruct the LM to respond as a specific persona, which we randomly sample from a large pre-specified set. Like seed prompting, persona prompting varies the prompt to steer toward varied outputs, but uses semantically meaningful personas rather than arbitrary strings.

Naive IID direct sampling. The bare task is sent as the user turn with no additional system prompt (creative writing tasks still receive the shared story guidance as the system turn). We draw n independent generations.

List prompting. We prompt the model to generate k complete responses as a numbered list. To get n samples, we perform $\lceil n/k \rceil$ total independent calls and then pool the parsed responses. We use $k = 8$ for creative writing and $k = 32$ for the chat and image domains. We arrived at these choices of k by extensive hyperparameter optimization, and find that it is critical for the list baseline's performance. Specifically, in initial experiments, we swept $k \in \{8, 16, 32, 64, 128, 256\}$ up to $n = 1024$ across many open-weight models and generally found that these specific choices of k worked best across models. Intuitively, the reason the creative writing domain uses a much lower value of k is because it demands much longer output—the LM cannot reason over many hundred-word stories at once. The system prompt is shown in Fig. 8.

List Baseline (system turn)

```
You are creative and can generate diverse, distinct outputs.

Generate {k} diverse responses to the task given by the user. Number
them 1-{k}. Each should be meaningfully distinct from the others, so
that the set of responses covers the full range of high-quality
possibilities.
```

Figure 8: System prompt for the list prompting baseline.

Verbalized Sampling. We follow the implementation of the original work Zhang et al. (2025a). A single call asks the model to verbalize a distribution of k responses and their corresponding model-estimated probabilities. We make $\lceil n/k \rceil$ independent calls to cover n samples. The probabilities are discarded after parsing. We use $k = 8$ for story and $k = 32$ for chat and image, which we optimize in the same manner as our list baseline above. For creative writing, we additionally include a response length constraint in the prompt with a target of 400 words, following the original work's creative writing setup. We find that without this constraint, the generated stories are single-sentence snippets as opposed to full stories. For the chat and image domains, this length constraint is omitted. The system prompt is shown in Fig. 9.

Direct Planning (Intent-Factored Generation) (Ahmed et al., 2025; Yao et al., 2019). We generate each response in two stages, implemented as two separate LM calls. First, we independently sample a plan or intent for the response at a higher-than-typical temperature (1.2). Then, in a second LM call, we execute the plan into a full response at the default temperature. With this method, diversity over responses comes from diversity over plans, which in turn comes from token sampling. The two-stage prompt is shown in Fig. 10.

Verbalized Sampling (system turn)

```
Generate {k} responses to the input prompt. [Each response should be
approximately {target_words} words.]
Return the responses in JSON format with the key: "responses" (list of
dicts). Each dictionary must include:
- 'text': the response string only (no explanation or extra text).
- 'probability': the estimated probability from 0.0 to 1.0 of this
response given the input prompt (relative to the full distribution).
Randomly sample the responses from the full distribution.
```

Figure 9: System prompt for the Verbalized Sampling baseline.

Intent-Factored Generation stage 1: brainstorm (system turn; temperature 1.2)

```
Brainstorm keywords or ideas for responding to the task given by the
user: the core content, angle, and approach a response could take. Give
only a short list of keywords or ideas | not the response itself.
```

Intent-Factored Generation stage 2: execute (system turn, then user turn)

```
[system]
Respond to the task given by the user, guided by the provided keywords
and ideas. Write the full response itself. Give the core answer only |
no preamble, no explanation, no meta-commentary, no labels or headers.
Strictly respect any length constraints in the task.

[user]
Task: {task}

Keywords and ideas: {plan}
```

Figure 10: Prompts for the two stages of Intent-Factored Generation: brainstorming a plan and executing it into a full response.

Persona prompting (Ge et al., 2024; Santurkar et al., 2023). We generate each response by instructing the LM to respond as if it were a specific persona (see Fig. 11), where the persona is uniformly sampled (without replacement, to ensure independent responses to the same prompt use different personas) from the `persona` column of the Nemotron-Personas-USA dataset (Meyer & Corneil, 2025).

Seed Prompting (String Seed of Thought) (Misaki & Akiba, 2026). We use the original work's prompt verbatim (see Fig. 12), in its seed-injection variant. For each sample, we generate a fresh 24-character random ASCII string. The specific vocabulary we draw over is all upper and lower-case letters, digits 0-9, and punctuation from `!@#$%^&*()-_=+`.

Shared prompt suffix for creative writing. To ensure that generation aligns with evaluation, every method (including Gacha Decoding, see Appendix B) is told which attributes the story diversity evaluation measures. For baselines on our creative writing domain, we append the following text to each baseline's system prompt (Fig. 13).

C.2 Evaluated Models

We serve all open-weight models on one H200 node with SGLang (Zheng et al., 2024). We present the models evaluated and sampling parameters used in evaluation in Table 5. In general, we use the default parameters recommended by the corresponding model card. We serve Inkling models at their native NVFP4, GPT-OSS models at their native MXFP4, and the remaining models at BF16.

Persona (system turn)

```
Persona:
{persona}

Produce one response to the task as it would be produced by the persona
above.

Use the persona to guide substantive choices: what possibilities are
noticed, what approach is taken, which details matter, what tradeoffs
are accepted, and how the response is structured.

Rules:

- Satisfy every explicit requirement in the task.
- Only use the persona to decide aspects that the task leaves open.
- The persona is the author or decision-maker, not necessarily a
character or subject in the response.
- Do not mention, describe, or explain the persona.
- Do not merely alter vocabulary, tone, or surface style.
- Commit to the response this persona would most naturally produce,
rather than trying to be generically comprehensive or broadly
appealing.
- Produce only the final response.
```

Figure 11: System prompt for persona prompting.

Seed Prompting (system turn)

```
You are a helpful AI Assistant designed to provide well-reasoned and
detailed responses. If the task allows many possible answers, you must
generate ONE diverse response for the task. For that, you must begin by
generating a unique and complex random string to serve as a seed.
This random string should appear sufficiently complex and
unpredictable, with no obvious structure or pattern. Use your judgment
to ensure it looks arbitrary and unguessable.
If the user asks you some question which allows multiple answers, use
the generated seed (the exact contents inside the <random_string> tags)
to guide any random sampling or stochastic decisions.
Follow these steps for every instruction:
1. Output the random seed string enclosed within <random_string> and
</random_string> tags.
2. Think deeply and carefully about the user's question, and enclose
this reasoning within tags. You have to generate ONE response
leveraging the generated seed---the exact contents inside the
<random_string> tags, to ensure your single answer is unique and
diverse. Make sure to extract maximum randomness from the string by
using all of its content.
3. Provide your final answer, enclosed within <answer> and </answer>
tags.
Strictly follow this tag structure, and respond in the following
format:
<random_string> ... </random_string> <answer> ... </answer>

A pre-generated random seed has been provided for you: {seed}
Do not generate your own random string. Use this exact seed as the
contents of the <random_string> tags, then follow the steps above to
derive your answer from it.
```

Figure 12: System prompt for seed prompting (String Seed of Thought), seed-injection variant.

Story guidance (appended to the system turn for creative writing tasks)

```
When generating diverse stories, give particular weight to varying them
along these axes: genre, setting, narrative tone, narrator and point of
view, premise, emotional resolution, dominant imagery, main character.
Do not override choices already fixed by the task.
```

Figure 13: Shared story-guidance suffix appended to the system prompt for creative writing tasks.

Model	Sampling parameters	Citation
Qwen 3.5 family	T=1.0, top-p=0.95, top-k=20, min-p=0.0, presence penalty 1.5; thinking on; 32,768 max tokens	(Qwen Team, 2026)
GPT-OSS family	T=1.0, top-p=1.0; reasoning effort medium; 65,536 max tokens	(Agarwal et al., 2025)
Inkling family	T=1.0, top-p=1.0; reasoning effort high; 65,536 max tokens	(Thinking Machines Lab, 2026)
Gemma 4 family	T=1.0, top-p=0.95, top-k=64; thinking on; 32,768 max tokens	(Team et al., 2026)
GPT 5.5	T=1.0 (API-fixed); default reasoning effort; 32,768 max tokens; `gpt-5.5`	(OpenAI, 2026)
Claude Opus 4.8	T=1.0 (API-fixed); default reasoning effort (adaptive, high); 32,768 max tokens; `claude-opus-4-8`	(Anthropic, 2026)

Table 5: Models used in our experiments with their sampling parameters. For open-weight models we adopt each model card's recommended generation settings with reasoning enabled; for API models we use the provider defaults (temperature=1.0).

C.3 Evaluation

We present details on evaluations for each domain, including any LM judge prompts used. We use Qwen3.6-27B as our LM judge with thinking on and its model-card recommended sampling parameters (same as the parameters for Qwen 3.5 models, see Appendix C.2). We use Qwen-3-Embedding-8B for text embeddings (Zhang et al., 2025b).

In-the-wild chat. We evaluate on the test split of InfinityChat (Jiang et al., 2026). We judge response validity with the prompt shown in Fig. 14.

Creative writing. We collect in-the-wild creative writing prompts by annotating and filtering Wildchat (Zhao et al., 2024) with an LM judge to keep only writing requests. From this subset, we perform a manual quality pass to remove ill-formed or illegible prompts, and uniformly sample 50 requests as our final evaluation set. A sample of 5 evaluation prompts is shown verbatim in Fig. 17.

We show the LM judge prompt we used to annotate stories along the semantic dimensions on which diversity is measured in Fig. 15. To score quality, we directly use the EQ-Bench Creative Writing judge prompt (Paech, 2023), following prior work (Zhang et al., 2025a). We show this prompt in Fig. 16 for reference.

Image generation. We filter OneIGBench for open-ended queries via an LM judge. Many of the prompts in OneIGBench are highly specific, and admit only a few valid images. We filter out these close-ended prompts in three stages. First, we exclude the text-rendering and knowledge-reasoning prompt categories, where the image content is essentially fully specified (e.g., `Generate the first page of the paper from the following information: Title: Denoising Diffusion Probabilistic Models...`). Next, we use an LM judge to annotate each prompt as open- or close-ended and remove all close-ended prompts. Finally, we pass all remaining prompts to the LM judge in a single context, and ask it to return the 50 most open-ended prompts. A selection is shown in Fig. 18.

Chat validity judge

```
You are grading whether a RESPONSE is a valid and complete answer to a
PROMPT.

The standard is a reasonable user's satisfaction: the RESPONSE is VALID
if a user who issued the PROMPT would be happy accepting it as an
answer -- it addresses everything they asked, is correct and coherent,
and honors every constraint they stated.

Guidelines:

1. Explicit constraints are binding. If the PROMPT states a requirement
   (a count, format, topic, source, length, etc.), the RESPONSE must
   honor it exactly. "Give me exactly three slogans" means exactly
   three; "a detective created by Agatha Christie" rules out Sherlock
   Holmes.

2. Unstated preferences are not constraints. If the PROMPT asks for a
   slogan and the RESPONSE offers several good ones, that is VALID --
   the user got what they asked for. Do not penalize length, style,
   tone, extra options, or extra explanation the user did not forbid.

3. Real things must be real. If the PROMPT asks to name or identify
   something that exists (in the real world or in real, published
   fiction), the RESPONSE must give a genuine instance; one invented by
   the responder is INVALID.

4. It must be a genuine, usable answer. A RESPONSE that is incoherent,
   off-topic, mostly placeholder, too incomplete to use, or that
   deflects the question back to the user without answering it is
   INVALID.

Respond with ONLY a JSON object: {"reason": "<one sentence>", "valid":
true|false}.
Prompt: {prompt}
Response: {response}
```

Figure 14: Validity-judge prompt for the chat domain.

Creative writing annotation judge prompt (system turn)

```
You are a careful literary analyst. You read a short story and describe
it along several dimensions. For each dimension, give a SHORT
DESCRIPTION--a few words to a single sentence--that captures what is
actually distinctive about THIS story on that dimension. Do not use a
fixed vocabulary or force the story into broad buckets: if two stories
differ subtly on a dimension, your descriptions should differ subtly
too. Base every description only on the text given; do not invent
details that are not there. If a dimension is genuinely absent or
indeterminate, say so briefly (e.g. "none" / "unclear"). Before
describing, FIRST judge whether the text is actually a complete short
story. Set "valid_story" to false if it is instead a one- or
two-sentence description/synopsis of a story, a bare outline or list of
beats, a refusal, empty/garbled output, or otherwise not a real
written-out narrative; set it to true only for an actual story.

The dimensions:
- genre: the story's genre or mode.
- setting: where and when it takes place (time, place, world).
- narrative_tone: the overall mood/voice.
- narrator: the narrative perspective and who tells it.
- premise: the core situation or what the story is fundamentally about,
in one phrase or sentence.
- resolution_emotion: the emotional note the ending lands on.
- dominant_imagery: the most prominent recurring images, motifs, or
sensory texture.
- main_character: who the central character is--a brief descriptor.

Respond with ONLY a JSON object. Its first key is "valid_story"
(boolean). Its remaining keys are exactly: genre, setting,
narrative_tone, narrator, premise, resolution_emotion,
dominant_imagery, main_character | each value is your description for
that dimension. No extra keys, no commentary outside the JSON.
```

Figure 15: The LM judge system prompt used to annotate stories along 8 semantic dimensions during diversity evaluation. The actual generated story is supplied as the user turn. The annotations are then embedded per-dimension (i.e., each story yields 8 dimensions $\rightarrow$ 8 embeddings), and we measure final diversity as average Vendi score across the dimensions. We adopt this setup following Springer et al. (2026).

Creative writing quality judge (EQ-Bench rubric)

```
You are an expert in assessing creative writing. Your task is to score the quality of the test model's
response below, by several metrics, on a 0-20 scale.

[PROMPT START]

{writing_prompt}

[PROMPT END]

[TEST MODEL RESPONSE]

{test_model_response}

[TEST MODEL RESPONSE END]

[Task]

You are an expert in assessing creative writing. Your task is to score the quality of the test model's
response below, by several metrics, on a 0-20 scale.

Scoring notes:

- A scores of 20 represents a masterpiece.

- If the criteria is not relevant to the piece (e.g. criteria relating to dialogue but the piece has none),
exclude it from your assessment.

- Do not penalize NSFW output; that is the default for some models. Just assess it on its merits.

- Everything within the "TEST MODEL RESPONSE" section was written by the test model. Sometimes models like
to write comments on the piece after the piece is concluded; if this happens you should ignore their
comments.

- You are to write a comprehensive analysis of the piece, then give your scores.

- If the output is not a story but rather a one or two sentence description or premise of a story, you
should score it very low on all criteria, as it is not a complete story.

- You are a critic, and your job is to be critical, especially of any failings or amateurish elements.

- After the analysis, output the scores as a SINGLE JSON object mapping each criterion's key to its integer
0-20 score. Output nothing after the JSON object.

- Output format is:

[Analysis]

Write your detailed analysis here.

[Scores]

{
  "surprising_creative": <0-20>,
  "imagery_descriptive": <0-20>,
  "nuanced_characters": <0-20>,
  "emotionally_complex": <0-20>,
  "elegant_prose": <0-20>,
  "earned_light_dark": <0-20>,
  "emotionally_engaging": <0-20>,
  "consistent_voice_tone": <0-20>,
  "natural_flow": <0-20>,
  "overall_engagement": <0-20>
}

---

Now, rate the supplied model output on the following criteria (use the JSON key shown for each):

Surprising and Creative  (key: "surprising_creative")
Imagery and Descriptive Quality  (key: "imagery_descriptive")
Nuanced Characters  (key: "nuanced_characters")
Emotionally Complex  (key: "emotionally_complex")
Elegant Prose  (key: "elegant_prose")
Well-earned Lightness or Darkness  (key: "earned_light_dark")
Emotionally Engaging  (key: "emotionally_engaging")
Consistent Voice/Tone of Writing  (key: "consistent_voice_tone")
Sentences Flow Naturally  (key: "natural_flow")
Overall Reader Engagement  (key: "overall_engagement")
```

Figure 16: Quality-judge prompt for the creative writing evaluation, from Paech (2023).

Creative writing task examples (5 of 50)

```
1. Tell me a story about three cats who went to Australia on a spring
break trip. Please make the story funny and full of valuable life
lessons.
2. Write a short story story about two neighbors
3. Write a story with moon idioms.
4. Write a story of a girl who likes to run
5. Write a story about someone who has no magic in a world of magic but
is so physically fit that he is stronger than many of those who can use
magic
```

Figure 17: Five prompts from our creative writing evaluation set (derived from in-the-wild user prompts in WildChat, typos preserved verbatim).

Image generation task examples (5 of 50)

```
1. A young child, seated at a lively restaurant table, eagerly observes
the vibrant, fragrant dishes of genuine, traditional cuisine being
served, captured in a lifelike style that showcases real people.
2. Swiss buildings, tall, deep underground view.
3. Aquarium expansion extends oceanward at sunset in futuristic long
beach.
4. On a rooftop overlooking a sprawling city, two casually dressed
individuals are seated. One is drinking from a cup, the other on phone.
5. Two characters hold hands; one dressed in a dark outfit with short
black hair, the other adorned in elaborate white and red armor. The
image unfolds in an artistic blend reminiscent of anime-style
storytelling.
```

Figure 18: Five randomly sampled prompts from our OneIG-Bench-derived image generation evaluation set (verbatim, including original phrasing).

When rendering images, we use Qwen-Image-2512 (Wu et al., 2025b) with 30 denoising steps, true-CFG scale 4.0, and at 1328×1328 resolution. We prompt the image model with "`{Original caption}\nExtra details: {LM-rewrite}`" as the positive prompt, and the default model-recommended negative prompt (which essentially just steers the model away from low-resolution outputs). We embed each image with DreamSim for diversity evaluation, using the best-performing ensemble model from the original work Fu et al. (2023).

D Details on protein design pilot study

D.1 Task overview

We compare Gacha Decoding with other inference-time methods for diversifying LM generations in a de novo protein binder design workflow. We task the LM to generate a set of diverse captions describing generic candidate binders with different folds and binding-interface configurations, and use a fixed text-conditioned protein generation model (Lee et al., 2026) to realize each caption in complex with some specified downstream target. We evaluate the resulting designs by measuring predicted interface confidence after refolding each binder sequence jointly with its target, and structural diversity among designs that pass an interface confidence threshold.

Target	Hotspot residues	Binder length
BHRF1	A65, A74, A77, A82, A85, A93	80–120
IFNAR2	A45, A73, A75, A77, A89, A91	60–175
PD1	A33, A95, A97, A102	80–150
TrkA	X294, X296, X333	50–120
PDL1	A37, A39, A49, A98	64–155
DerF21	A10	70–185
IL7RA	B58, B80, B139	50–120
VEGFA	W81, W83, W91	50–140

Table 6: Targets considered in our protein binder design pilot (Pacesa et al., 2025). Hotspots are specified by chain ID and residue number in the input structure. Binder lengths are sampled uniformly from the ranges shown (inclusive).

We consider eight benchmark targets following Pacesa et al. (2025): BHRF1, IFNAR2, PD1, TrkA, PDL1, DerF21, IL7RA, and VEGFA. For each target, we use the sequence and structure, predefined hotspot residues specifying the intended binding site, and binder-length range supplied by the benchmark configuration (Table 6 for reference). We sample binder lengths uniformly from the target-specific range; these target-specific inputs are then supplied as conditioning to the protein generation model alongside the generated captions. Note that we reuse the same length draws across caption generation methods to ensure fair comparison. Also note that these target specific inputs are not specified to the caption-generating LM—initial results showed that across methods, omitting this information led to better performance. All methods receive the same shared captioning task prompt, shown in Fig. 19, which includes few-shot examples.

Using this prompt, we generate binder captions with GPT-5.5, one of the two frontier models among the models we considered. In preliminary experiments, Opus 4.8 often declined requests involving biological design. We compare Gacha Decoding with direct sampling, list prompting, and verbalized sampling, whose method-specific system instructions are detailed in Appendix C.1. Direct sampling provides a naive reference, and list prompting and verbalized sampling were generally the best baselines on GPT 5.5 in our main experiments. We generate 1024 captions with each method.

Method hyperparameters. Following our main experiments, we use $k = 32$ for list prompting and verbalized sampling. For Gacha Decoding, we use 7 decision steps and prompt for 16 enumerated options per decision.

D.2 Evaluation of binder captions

Given each generated caption, we pair it with a binder length sampled from the target-specific binder-length range, the target's sequence and structure, and a fixed set of hotspot residues defining

Protein binder design task

```
Generate a caption describing a designed protein binder: a single-chain
protein that folds on its own and binds a target protein at a defined
epitope on that target's surface. The binder's CATH class must be
either mainly alpha or alpha beta. Describe the binder itself -- its
fold and the interface it forms with the target. Do not name a target
protein, do not specify which epitope is bound, and do not state the
binder's length; those are fixed by the design task.

Each caption should be factual and descriptive, using concrete numbers
and specific structural terms. Write each caption as a bulleted list,
one attribute per line, in the form '- <attribute>: <value>', using the
attribute names from the category list verbatim. No speculative
language, no prose paragraphs, no commentary.

Example captions for reference style (match this format exactly):
1.
- CATH class: mainly alpha
- CATH architecture: up-down bundle
- helix count: 3
- sheet count: 0
- compactness and shape: compact globular
- surface hydrophobicity level: low outside the interface
- interface residue count: 24
- interface buried surface area: 890 square Angstrom
- interface hydrophobic contact count: 17
- interface H-bond count: 4
- interface geometry type: flat helical face
- interface shape complementarity score: 0.71

2.
- CATH class: alpha beta
- CATH architecture: two-layer sandwich
- helix count: 2
- strand count: 5
- sheet count: 1
- beta topology type: antiparallel meander
- contact order value: 18
- disulfide bond count: 1
- disulfide topology type: intra-chain
- interface residue count: 31
- interface buried surface area: 1140 square Angstrom
- interface H-bond count: 11
- interface salt bridge count: 3
- interface geometry type: concave beta face wrapping a protruding loop
- interface shape complementarity score: 0.66
```

Figure 19: Shared task prompt for protein binder caption generation. Target-specific inputs are supplied to the protein generation model; each method's caption set is reused across all eight targets.

the desired epitope, and pass these inputs to a fixed text-conditioned protein structure generation model (Lee et al., 2026). The target is represented as residue tokens carrying its sequence and structural features, while the hotspot mask enters through target sequence and pair features to identify residues the binder should contact. The caption is encoded through two complementary frozen language-model pathways: Qwen-3-Embedding-8B produces a pooled representation that is projected and broadcast to every binder residue through adaptive layer normalization, providing global conditioning, while Qwen-3-8B produces token-level representations that the protein states attend to in each denoiser layer, preserving finer-grained structural and interface constraints. Starting from Gaussian noise for the binder's C_α coordinates and per-residue side-chain latents, the model keeps the target fixed and generates the binder by simulating its learned stochastic differential equation for 400 Euler–Maruyama steps with classifier-free guidance. A frozen variational autoencoder then decodes the sampled representation into an amino acid sequence and all-atom target–binder complex. We generate one complex per caption while holding the target, hotspots, binder-length range, and sampling configuration fixed across caption generation methods.

To evaluate predicted binding, we refold each generated binder sequence jointly with its target using ESMFold2-Fast (Candido et al., 2026) and compute $\text{ipSAE}_{\min}$ from the resulting predicted aligned error matrix (Dunbrack Jr, 2025). For each residue pair spanning the predicted binder–target interface, ESMFold2 estimates the error in their relative positions; ipSAE converts these errors into a score between 0 and 1, with low predicted error yielding a high score, and adjusts the normalization for interface size. $\text{ipSAE}_{\min}$ takes the smaller of the asymmetric binder-to-target and target-to-binder scores, so a high value requires confident inter-chain geometry in both directions and serves as a proxy for predicted binding strength rather than a direct affinity measurement. We operationally designate designs with $\text{ipSAE}_{\min} > 0.5$ as successful. To measure diversity among successful designs, we remove the shared target chain and compare the generated binders within each target using all-vs-all Foldseek with exact TM-align (Van Kempen et al., 2024; Zhang & Skolnick, 2004). TM-score measures global structural similarity after optimal three-dimensional alignment, normalized for protein length: it ranges from 0 to 1, with larger values indicating more similar folds and 1 indicating identical structures. We symmetrize the pairwise TM-scores and use the resulting similarity matrix as the Vendi kernel. The Vendi score summarizes the spectrum of this matrix and can be interpreted as the effective number of structurally distinct binders, such that higher values indicate greater structural diversity. We compute diversity separately for each target so that differences between target structures or epitopes do not contribute to the score.