"Trust Yourself": Unsupervised Self-Evolution of Reasoning through Model-Intrinsic Verification

Anonymous ACL submission

Abstract

Self-evolving reasoning seeks to improve large language models (LLMs) through iterative feedback, yet existing approaches often rely on external supervision or rigid memory mechanisms, limiting scalability in truly unsupervised settings. We propose **SEER**, an *unsupervised* **S**elf-**E**volution framework for **E**xperience-aware **R**easoning that enables LLMs to self-improve using only intrinsic verification signals. SEER treats reasoning as a sampling process and employs MCMC-based exploration to generate diverse reasoning trajectories. It leverages *exploration consensus*, the agreement across sampled trajectories, to identify inputs with meaningful learning potential, and distills compact key-point experiences from these trajectories. Candidate experiences are then validated through *experience confidence*, which measures whether injecting an experience increases decision stability and reduces uncertainty. Only experiences that pass this intrinsic, label-free verification are stored in a persistent memory bank. During inference, SEER dynamically retrieves experiences only when the model's confidence is low, enabling selective and low-noise memory injection. Extensive experiments show that SEER consistently outperforms strong reasoning baselines, highlighting SEER as a practical step toward scalable, training-free, unsupervised self-evolution in LLMs.

1 Introduction

Large language models (LLMs) are increasingly shifting from one-shot decoding to iterative reasoning—a paradigm in which solutions are refined through multi-step deliberation, revision and self-correction (Zhan et al., 2025; Li et al., 2025a). This shift has enabled stronger performance on complex tasks, and has motivated a growing body of work that seeks continuous improvement without retraining the entire model, e.g., via iterative optimization (Shinn et al., 2023) or by reusing past trajectories through context-based memory (Ouyang et al., 2025). The long-term frontier of this line of research is self-evolution: an open-ended mechanism where the model improves itself through recursive feedback loops, gradually accumulating reusable knowledge and adapting beyond a single task distribution (Wu et al., 2025a; He et al., 2025b; Acikgoz et al., 2025).

Despite encouraging progress, existing self-evolving frameworks remain limited by two structural constraints: reliance on external supervision or rigid retrieval strategies. Training-based methods (Zhao et al., 2025; Yan et al., 2025; Wu et al., 2025b; Wang et al., 2025a) typically depend on explicit reward signals to drive reinforcement learning or policy optimization. In open-ended domains, however, such rewards are often sparse, ambiguous, or expensive to construct, making supervision a fundamental bottleneck and hindering scalability to genuinely unsupervised settings (Yue et al., 2025; He et al., 2025a). Inference-only alternatives (Liu et al., 2025) avoid gradient updates but lack principled mechanisms for experience validation and adaptive usage. Without intrinsic signals to assess experience quality or model confidence, these approaches either store noisy raw trajectories or over-compress them, and typically retrieve context indiscriminately for every query. Such static retrieval not only wastes computation but can also degrade performance by injecting irrelevant information into otherwise simple reasoning tasks.

This work aims to study a question: *Can LLMs self-evolve in a fully unsupervised manner, using only intrinsic signals to (i) explore better reasoning paths, (ii) verify which experiences are beneficial, and (iii) retrieve them only when needed?*

To this end, we propose **SEER**, an unsupervised **S**elf-**E**volution framework for **E**xperience-aware **R**easoning. SEER eliminates the reliance on gold labels, human feedback, and reward models. In-

stead, it relies on a set of intrinsic verification signals available within any LLM, which capture both the *exploration consensus* among its reasoning outcomes and its *experience confidence* during improved generation. These internal signals form a closed feedback loop that enables self-evolution without external supervision.

SEER instantiates a closed-loop, fully unsupervised self-evolution pipeline with three coupled components: **(1) MCMC-based trajectory exploration** to discover diverse, high-quality reasoning trajectories without external rewards; **(2) experience distillation and filtering** to extract compact key-point experiences and retain them only when they increase exploration consensus and improve experience confidence; and **(3) dynamic entropy-aware gating** to retrieve and inject the verified experiences only under sustained uncertainty, avoiding unnecessary overhead on easy instances. Together, these modules enable unsupervised self-correction, compact experience representation, and adaptive memory usage within a single training-free framework.

We conduct extensive experiments to evaluate the effectiveness of SEER on mathematical reasoning, code generation and domain expertise benchmarks using Qwen2.5 and Qwen3 series models. The results show that our method achieves significantly improved reasoning performance and versatility across diverse domains. Further analyses indicate that our method can universally enhance reasoning capabilities across different model scales without compromising performance stability.

2 Related Work

Self-Evolving Reasoning Recent work frames self-evolving reasoning primarily as an inference-time process. Leading search-based methods, such as LATS (Zhou et al., 2024) and DSER (Liu et al., 2025), integrate verbal reinforcement and Monte Carlo Tree Search to actively prune reasoning paths without parameter updates. Sustaining such evolution requires robust context engineering to optimize information processing across three dimensions: collection via vector-based memory (Wu et al., 2022); management for hierarchical abstraction (e.g., MemOS (Li et al., 2025b)); and usage through Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) and multi-agent coordination (Qian et al., 2024). This synergy culminates in memory-centric frameworks like ReasoningBank (Ouyang et al., 2025) and Memory-R1 (Yan et al., 2025), which leverage managed trajectories to guide future inference. Complementarily, training-time self-evolution represents another critical frontier, where frameworks like AbsoluteZero (Zhao et al., 2025) bootstrap capabilities via reinforcement learning and self-play (Wu et al., 2025b). However, the critical limitations persists: training methods are largely constrained by the bottleneck of external supervision in open-ended domains, while inference-centric approaches struggle with rigid retrieval strategies and a lack of intrinsic mechanisms to validate the quality of accumulated experiences.

Entropy as a Reasoning Signal Moving beyond its traditional role as a static uncertainty metric, entropy is increasingly reconceptualized as a dynamic reasoning signal that reveals the model's internal cognitive state. Lee et al. (2025) identify that entropy fluctuations correlate linearly with the perception of task difficulty. In this context, high-entropy regions are not merely noise, but are identified by Cheng et al. (2025) as critical junctions where the model engages in logical branching or self-verification. Crystallizing this perspective, Wang et al. (2025b) posit that these moments constitute a "high-entropy minority" that carries the densest reasoning information. They demonstrate that while the majority of tokens are redundant, these high-entropy signals pinpoint exactly where the model attempts to bridge logical gaps, acting as the primary driver for acquiring and refining complex reasoning behaviors.

3 Method

In this section, we propose **SEER**, a fully *unsupervised* **S**elf-**E**volution framework for **E**xperience-aware **R**easoning in LLMs without any external supervision, including gold labels, human feedback, and reward models. Unlike typical methods that rely on curated annotations or learned evaluators, SEER is driven entirely by the model's intrinsic verification signals: (i) *exploration consensus*, capturing the agreement of reasoning outcomes across sampled trajectories, and (ii) *experience confidence*, reflecting the model's internal certainty during generation. These intrinsic signals serve two roles: they guide exploration toward diverse yet plausible reasoning paths, and they validate whether distilled heuristics genuinely stabilize the model's decisions. Figure 1 illustrates

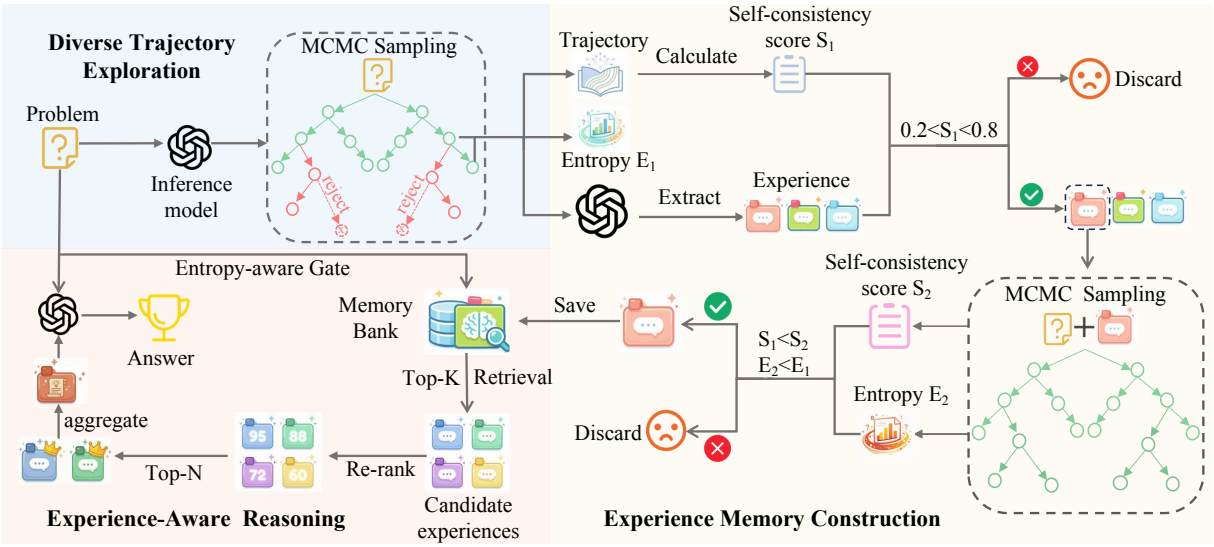

Figure 1: Overview of the SEER framework. The pipeline consists of three key phases: (1) Diverse Trajectory Exploration via MCMC sampling to generate diverse reasoning paths; (2) Experience Memory Construction utilizing dual-stage filtration and dynamic management to distill valid experiences; and (3) Experience-Aware Reasoning which retrieves and utilizes stored experiences when inference uncertainty is high.

its overall architecture.

3.1 Diverse Trajectory Exploration

Standard decoding strategies (e.g., greedy search or temperature sampling) often under-explore the solution space, leading to limited reasoning diversity and mode collapse. Following the "reasoning as sampling" perspective (Karan and Du, 2025), we treat reasoning generation as sampling from a sharpened distribution induced by the base model. Importantly, this exploration remains fully unsupervised: it uses only the base model likelihood and does not require any external reward signal. Let x denote the input query and $z = (z_1, \ldots, z_T)$ be a reasoning trajectory. We define the target power distribution as follows:

$$\pi(z \mid x) \propto P_\theta(z \mid x)^\alpha, \tag{1}$$

where $\alpha \geq 1$ controls distribution sharpening. Larger α upweights high-likelihood trajectories while still allowing diverse exploration over multiple modes. A theoretical comparison to temperature sampling is provided in Appendix A.1.

Direct sampling from π is intractable due to the unknown normalization constant. We therefore apply Metropolis-Hastings (MH), using a proposal distribution $q(z'|z_t)$ instantiated by the base model P_θ with an appropriate temperature (e.g., $\tau = 1/\alpha$). Given current state z_t, a proposed trajectory z' is accepted with probability:

$$A(z', z_t) = \min\left(1, \frac{P_\theta(z' \mid x)^\alpha \cdot q(z_t \mid z')}{P_\theta(z_t \mid x)^\alpha \cdot q(z' \mid z_t)}\right). \tag{2}$$

Iterating this procedure yields a set of diverse reasoning trajectories $\mathcal{T} = \{z^{(1)}, \ldots, z^{(K)}\}$ that reflect multiple plausible plausible solution modes. The full procedure is formally summarized in Algorithm 1 at Appendix A.2.

3.2 Experience Memory Construction

The trajectories are transient and usually contain instance-specific artifacts. SEER thus converts these trajectories into a experience memory bank, which stores generalizable reasoning key point experiences. Crucially, experience acquisition is performed in a label-free manner: a candidate experience is accepted only if it measurably improves the model's intrinsic verification signals, namely exploration consensus and experience confidence.

3.2.1 Trajectory-to-Experience Distillation

MCMC sampling yields an evolving trajectory chain $\mathcal{Z}_x = \{z^{(0)}, \ldots, z^{(K)}\}$. However, directly storing raw trajectories is undesirable, as they inevitably conflate reusable reasoning strategies with instance-specific content (e.g., numbers, entity names, surface wording). We therefore introduce contrastive key-point distillation that extracts a minimal and generalizable procedural update from consecutive MCMC states.

Contrastive Key-Point Distillation. Given two adjacent trajectories $z^{(t-1)}$ and $z^{(t)}$, we construct a deterministic contrastive input $\Delta(z^{(t)}, z^{(t-1)}) \triangleq \textsc{Pack}(x, z^{(t-1)}, z^{(t)})$, where $\textsc{Pack}(\cdot)$ concatenates the problem x and the two trajectories with separators, exposing what changed between them. We then extract key-point experiences:

$$k^{(t)} = \Phi_{\text{extract}}\left(\Delta(z^{(t)}, z^{(t-1)}) \mid x, \rho_{\text{contrast}}\right), \quad (3)$$

where ρ_{contrast} is a contrastive meta-instruction that asks for the minimal procedural update turning $z^{(t-1)}$ into $z^{(t)}$. Φ_{extract} is implemented by the same base LLM under a teacher prompt role (or an identical frozen copy), requiring no labels, human feedback, or reward models.

Why contrasting consecutive states helps? This design acts as a semantic sieve. Adjacent trajectories typically share the same instance-specific details (e.g., constants, entity names) while differing primarily in their reasoning operations (e.g., decomposition choice, constraint handling, or error correction). By distilling from the differential signal $\Delta(z^{(t)}, z^{(t-1)})$ rather than from a single trajectory in isolation, the extractor is encouraged to ignore static problem-specific content and focus on the reasoning shift. As a result, the memory bank stores generalized experiences that are more likely to transfer across inputs, instead of memorizing the particulars of the current problem.

3.2.2 Dual-Stage Experience Filtering

A distilled experience k_{cand} is not guaranteed to be correct or transferable. Moreover, experiences extracted from different problems can vary drastically in signal quality: some problems are too hard (model outputs are essentially random), while others are too easy (the model is already stable), both of which provide little useful learning signals. To avoid noisy experiences, SEER adopts a dual-stage, label-free filtering pipeline driven only by intrinsic verification signals: *exploration consensus* and *experience confidence*, which are operationalized via self-consistency (SC) and predictive entropy, respectively.

Stage 1: Potential Filtering (input-level). This stage filters experiences indirectly by selecting which problem inputs are worth learning from. Given an input x and its reasoning trajectories from MCMC sampling, we compute the baseline exploration consensus score, operationalized via self-consistency, denoted as S_1 (details in Appendix B). We keep only those inputs whose baseline consistency falls into an empirical "learning zone": $0.2 < S_1 < 0.8$. All experiences distilled from inputs outside this interval are discarded. The motivation is to selectively focus memory construction on informative cases: when $S_1 \leq 0.2$, trajectories are highly divergent with no consensus, so extracted experiences are likely dominated by noise or lucky guesses; when $S_1 \geq 0.8$, the model already converges to a stable solution mode, so extracted experiences tend to be redundant and offer limited marginal gain. By restricting to intermediate-consistency inputs, we prioritize problems where the model is uncertain yet not random, thereby yielding higher-quality and more generalizable experiences.

Stage 2: Efficacy Verification (experience-level). Inputs passing Stage 1 only indicate potential learnability; they do not guarantee that a specific k_{cand} is helpful. Stage 2 thus performs a *counterfactual self-test* on the same input x to assess the experience confidence by comparing the model states *before* and *after* injecting k_{cand}. Let (S_1, E_1) be the baseline SC score and average entropy computed from the original generation (no injection), and let (S_2, E_2) be the corresponding metrics after injecting k_{cand} into the context and re-sampling (entropy details in Appendix B). We accept k_{cand} into the memory bank $\mathcal{B}$ if and only if $S_2 > S_1$ and $E_2 < E_1$. The first condition enforces improved exploration consensus across trajectories, while the second requires increased experience confidence during generation, jointly filtering out candidates that merely shift outputs without providing reliable guidance. This dual criterion enables SEER to validate experiences without any external labels or reward models, committing only those that yield measurable intrinsic improvements.

3.2.3 Experience Memory Maintenance

The memory bank $\mathcal{B} = \{e_i\}_{i=1}^{M}$ stores entries as key-value pairs: a query representation q_i (source problem context) and a verified experience k_i, together with lightweight metadata (e.g., access counts and timestamps). To avoid redundancy, we apply semantic deduplication at insertion time: if the cosine similarity between k_{cand} and an existing k_i exceeds a threshold δ, we overwrite the old entry; otherwise we create a new entry.

To maintain relevance under continual updates,

we periodically prune the bank every T_{prune} insertions using a utility score:

$$U(e_i) = N_{\text{hit}}(i) \cdot \exp\left(-\lambda \cdot (t_{\text{curr}} - t_{\text{last}}^{(i)})\right), \quad (4)$$

where $N_{\text{hit}}(i)$ is the access frequency of entry e_i, λ is a time-decay coefficient, t_{curr} denotes the current time step, and $t_{\text{last}}^{(i)}$ represents the time step of the most recent access to entry e_i. Entries with the lowest utility are removed, enabling $\mathcal{B}$ to evolve toward robust and frequently useful experiences. The storage schema is detailed in Appendix D.

3.3 Experience-Aware Reasoning

Standard retrieval-augmented generation (RAG) pipelines typically retrieve for every query, which often (i) wastes tool/context budget on easy instances and (ii) introduces irrelevant or even misleading memories due to purely lexical matching. SEER instead treats memory as optional assistance: it is invoked only when the model is genuinely uncertain and the retrieved experiences are further filtered for structural alignment and functional utility. This design keeps inference lightweight on confident cases while providing targeted guidance when uncertainty persists.

Uncertainty-based Adaptive Gating. We estimate generation uncertainty using smoothed Shannon entropy $\bar{H}_t$, computed as a moving average over a local sliding window (Appendix C). Compared to raw token-level entropy (and the entropy used in B.2), $\bar{H}_t$ captures sustained uncertainty rather than transient fluctuations, making it a more reliable trigger for memory usage. We then define an adaptive gate $\mathbb{G}$ controlled by a threshold γ:

$$\mathbb{G} = 1 \iff \bar{H}_t > \gamma.$$

When $\mathbb{G} = 0$, the model continues decoding purely with its parametric knowledge. When $\mathbb{G} = 1$, the model retrieves experiences from the memory bank and conditions its subsequent reasoning on the retrieved guidance. In this way, SEER avoids "always-on retrieval" and instead performs uncertainty-driven memory invocation.

Memory Reranking and Synthesis. Once the gate is activated, directly retrieving by embedding similarity can still erroneously return experiences that are textually similar but logically mismatched. To ensure that injected memories are both *isomorphic to the current problem* and *useful for the current reasoning state*, we adopt a systematic three-step retrieve-rerank-synthesize pipeline:

(1) Problem-Anchored Retrieval. We first retrieve a candidate set $\mathcal{M}_{\text{cand}}$ of top-K experiences using problem-level similarity. Each memory entry stores the embedding of its source problem $E(q^{(i)})$. Given the current input x, we compute cosine similarity between $E(x)$ and $E(q^{(i)})$ and retrieve the top-K entries. This problem-anchored strategy prioritizes structural similarity (problem isomorphism) rather than superficial overlap between the current query and the experience text.

(2) Utility-Aware Reranking. Even structurally similar experiences may be irrelevant to the model's current partial solution. We therefore rerank candidates by state-conditioned utility. Specifically, we prompt the model to act as a scorer and assign each candidate a relevance score s_i based on its expected helpfulness to the current partial generation $y_{<t}$. We then select the top-N experiences ($N < K$) with the highest s_i, filtering out incompatible candidates.

(3) Dynamic Synthesis. Naively concatenating multiple experiences can fragment context and introduce redundancy or conflicts. We thus perform an in-context synthesis step that rewrites the selected top-N experiences into a single coherent instruction block I_{syn}. This synthesis removes duplicated advice, resolves inconsistencies, and yields a compact guidance prompt that is easier for the model to follow, thereby stabilizing subsequent decoding and reducing uncertainty.

3.4 Comparison to Previous Work

Table 2 presents our comparisons to prior work in three key aspects. First, SEER establishes a fully unsupervised self-evolution paradigm. Unlike Absolute Zero (Zhao et al., 2025), Memory-R1 (Yan et al., 2025), and Socratic-Zero (Wang et al., 2025a), which rely on external reward models (teacher model), or EvolveR (Wu et al., 2025b) which relies on ground-truth annotations to drive updates, SEER requires no external supervision.

Second, distinct from them that utilize synthetic questions and reasoning trajectories (Zhao et al., 2025), contrastive positive and negative candidate pairs (Wang et al., 2025a), or solutions verified against ground truth (Wu et al., 2025b) as evolution sources, SEER innovatively leverages the model's intrinsic verification signals to evolve purely from self-distilled reasoning experience.

Finally, SEER adopts a training-free memory construction mechanism. In contrast to training-based approaches (e.g., EvolveR, Memory-R1)

Method	Math				Code		Knowledge	
	GSM8K	MATH500	AIME24	AIME25	HumanEval	MBPP	GPQA	MMLU
Qwen2.5-1.5B								
CoT SC	68.7	34.2	-	-	43.3	60.7	26.5	67.0
ToT	64.2	30.4	-	-	42.1	57.8	22.5	53.6
LATS	65.0	27.6	-	-	41.5	58.6	24.3	57.3
Reflexion	70.4	36.2	-	-	43.9	60.2	26.1	62.4
DSER	70.0	31.2	-	-	43.3	63.9	26.5	61.7
SEER (Ours)	**72.9**	**44.6**	-	-	**58.5**	**67.3**	**28.1**	**67.5**
Qwen2.5-7B								
CoT SC	85.8	50.2	3.33	2.50	61.0	75.4	38.1	75.2
ToT	83.7	42.2	0.00	0.00	59.1	71.4	30.3	63.8
LATS	84.2	41.6	0.83	0.00	60.4	72.8	28.5	64.8
Reflexion	85.3	51.6	1.67	1.67	62.2	74.5	34.1	68.5
DSER	84.5	50.4	2.08	1.67	61.6	74.6	36.3	71.7
SEER (Ours)	**88.4**	**60.8**	**5.83**	**5.00**	**68.9**	**78.7**	**40.1**	**76.8**
Qwen2.5-32B								
CoT SC	92.9	58.4	6.67	4.17	64.0	85.7	48.6	83.7
ToT	91.6	56.8	1.67	1.25	62.2	81.6	36.7	69.3
LATS	91.8	51.6	2.50	1.67	64.6	83.9	37.6	70.3
Reflexion	92.0	61.4	3.33	4.17	63.4	84.1	41.6	73.3
DSER	92.1	59.2	3.33	3.33	65.2	83.9	43.4	67.4
SEER (Ours)	**93.1**	**66.2**	**9.17**	**7.50**	**76.8**	**88.2**	**50.1**	**84.0**
Qwen3-8B								
CoT SC	90.8	61.0	15.83	10.42	63.4	71.2	45.2	78.4
ToT	89.4	58.8	11.67	6.67	62.2	68.6	34.5	65.8
LATS	90.7	55.8	12.08	7.50	63.4	69.1	33.6	66.9
Reflexion	91.3	61.6	13.75	8.75	64.0	69.9	35.4	70.3
DSER	92.7	58.6	12.91	9.58	65.8	70.8	36.3	73.0
SEER (Ours)	**96.4**	**67.4**	**17.92**	**15.83**	**73.2**	**74.6**	**46.1**	**79.2**
Qwen3-14B								
CoT SC	92.5	62.2	17.08	14.58	62.2	74.6	40.3	81.4
ToT	89.9	61.0	12.50	10.83	60.4	69.0	32.1	68.5
LATS	90.0	59.6	13.33	11.67	61.6	69.9	30.7	69.6
Reflexion	90.3	62.2	14.58	13.75	62.8	71.4	34.1	73.0
DSER	90.8	60.8	15.42	12.91	64.6	73.6	34.7	77.3
SEER (Ours)	**92.8**	**69.6**	**19.17**	**18.33**	**70.1**	**76.3**	**42.3**	**81.9**

Table 1: Main results across math reasoning, code generation, and domain expertise tasks. Best results are bolded.

that require expensive gradient-based parameter optimization, SEER avoids this computational burden. While DSER is also training-free, its evolution source is primarily based on transient exploration rollouts, which treats reasoning as a temporary Markov chain. SEER, conversely, focuses on accumulating long-term retrievable experience, offering better generalization than transient rollouts

Method	Supervision	Evolution Source	Training-Free
SEER (Ours)	None	Experience	✓
DSER	None	Exploration Rollouts	✓
Absolute Zero	RM	Synthetic Data	✗
Memory-R1	RM	Synthetic Data	✗
Socratic-Zero	RM	Contrastive Pairs	✗
EvolveR	GL	Verified Data	✗

Table 2: Comparison to prior work. "RM" and "GL" denote reward models and gold labels, respectively.

4 Experiments

4.1 Experimental Setup

Benchmarks and Metrics. We evaluate SEER on three domains to assess generality: **(1) Mathematical reasoning:** GSM8K (Cobbe et al., 2021), MATH500 (Lightman et al., 2023), AIME24 (Zhang and Math-AI, 2024), and AIME25 (Zhang and Math-AI, 2025). For the AIME benchmarks, we report the mean accuracy over 8 runs to account for variance; we exclude the 1.5B model on AIME where it achieves 0% accuracy. **(2) Code generation:** HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). **(3) Domain expertise:** GPQA (Rein et al., 2023) and MMLU (Hendrycks et al., 2021), covering expert-level knowledge in biology, physics, and chemistry. Unless otherwise specified, we report standard task metrics (e.g., accuracy / exact match for QA-style tasks and pass@1 for code generation).

Baselines. To comprehensively evaluate SEER, we benchmark it against five representative state-of-the-art methods: CoT-SC (Wang et al., 2023), which employs majority voting over multiple sampled paths; tree search methods including ToT (Yao et al., 2023) and LATS (Zhou et al., 2024), which utilize lookahead planning and Monte Carlo

Model	GSM8K			MATH500			AIME24			AIME25		
	Baseline	SEER	Δ	Baseline	SEER	Δ	Baseline	SEER	Δ	Baseline	SEER	Δ
General Purpose Models												
Qwen2.5-7B-Base	85.8	88.4	+2.6	49.8	60.8	+11.8	3.33	5.83	+2.50	2.50	5.00	2.5
Qwen2.5-7B-Instruct	89.3	91.2	+1.9	75.6	77.8	+2.2	13.3	17.5	+4.20	9.17	13.33	4.16
Math Specialized Models												
Qwen2.5-Math-7B	89.6	91.7	+2.1	55.4	67.6	+12.2	4.17	9.17	+5.00	3.33	9.17	+5.84
Qwen2.5-Math-7B-Instruct	90.0	93.5	+3.5	78.2	81.5	+3.3	13.33	14.17	+0.84	10.00	14.17	+4.17

Table 3: Versatility analysis of SEER across different Qwen2.5 variants.

Tree Search; and self-correction approaches like Reflexion (Shinn et al., 2023) and DSER (Liu et al., 2025), which focus on verbal reinforcement and iterative self-evolution, respectively.

Implementation Details. To ensure reproducibility, we utilize the open-weights **Qwen2.5** and **Qwen3** series as base models, while employing GPT-4o as the score model to get relevance score during inference in Section 3.3 and the extractor for generating high-quality reasoning experiences during memory distillation. We configure the MCMC process with $N = 15$ steps and temperature $\tau = 0.25$ to ensure sufficient exploration, limiting distilled experiences to 1024 tokens. For adaptive inference, we employ all-MiniLM-L6-v2 as the embedding model, with the gating threshold γ empirically optimized on the validation set.

4.2 Main Results

Effectiveness. As presented in Table 1, SEER consistently yields superior performance across diverse domains, outperforming both standard prompting and advanced inference-time search baselines. In mathematical reasoning, the method demonstrates remarkable effectiveness on complex tasks. For instance, it boosts Qwen3-14B to 18.33% on the rigorous AIME25 benchmark and improves Qwen2.5-32B to 66.2% on MATH500. Furthermore, SEER exhibits strong capabilities: on code generation, Qwen2.5-32B achieves a substantial jump to 76.8% on HumanEval, while also setting new highs on knowledge-intensive benchmarks like MMLU (84.0%).

Versatility. To comprehensively assess the adaptability of our framework, we extend our evaluation to four distinct variants of the Qwen2.5-7B series. As illustrated in Table 3, for the Base models which typically lack instruction-following optimization, SEER unlocks dormant reasoning potential, yielding substantial gains of up to 12.2% on MATH500. Furthermore, even on the highly specialized Math-Instruct model, SEER breaks through the performance ceiling, contributing an additional 4.17 points on the challenging AIME25 benchmark. These results confirm that SEER acts as a versatile enhancer, providing gains that are orthogonal to standard supervised fine-tuning.

Stability. Given the dynamic nature of our self-evolving memory, the chronological order of queries could theoretically bias the accumulated experience. To rule out such order sensitivity, we conducted a rigorous stability test by performing 10 independent runs on MATH500 with randomly shuffled dataset sequences. As visualized in Figure 3, SEER exhibits exceptional stability across all model scales. Specifically, Qwen2.5-32B maintains a steady performance with an average accuracy of 66.20%, where fluctuations are strictly confined to a negligible range (65.8%–66.6%). Similarly, the 7B and 1.5B models converge to stable averages of 60.80% and 44.60% respectively, exhibiting minimal variance across runs.

4.3 Ablation Studies

To rigorously evaluate the contribution of each component within the SEER framework, we conducted a systematic ablation study across different model sizes. The results, presented in Table 4, offer several critical insights. First, replacing MCMC sampling with standard temperature sampling results in a notable performance drop (e.g., -5.2% with Qwen2.5-1.5B on MATH500). This underscores the necessity of MCMC's acceptance-rejection mechanism in navigating complex solution spaces. Second, removing experience distillation, i.e., using raw reasoning traces directly, leads to a decline in accuracy. Raw trajectories often contain redundant or irrelevant tokens that dilute the context window, whereas distilled experiences provide focused, high-density guidance. Finally, eliminating the entropy gate leads to a significant

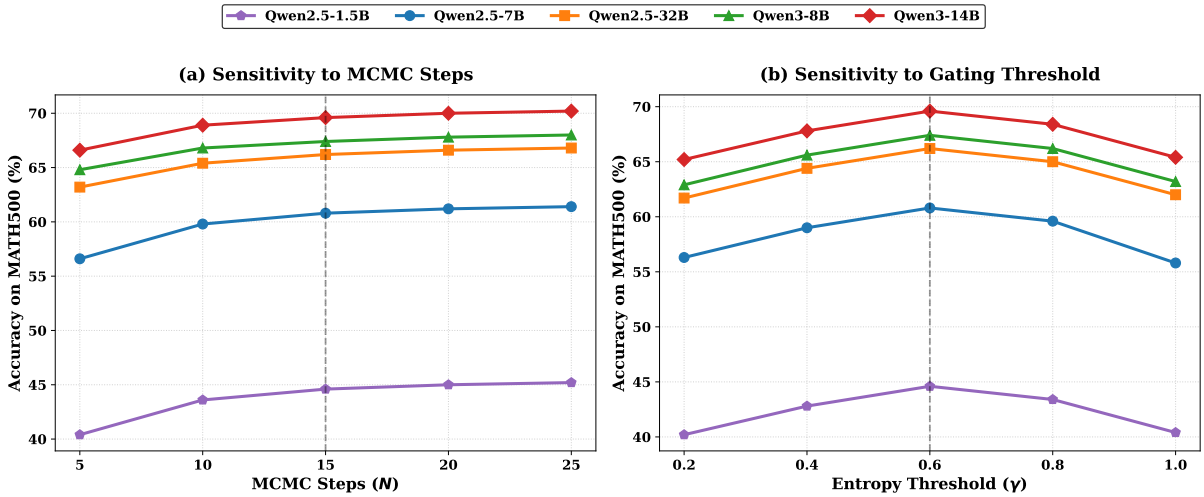

Figure 2: Sensitivity analysis of reasoning performance on MATH500 across five model scales with respect to MCMC exploration steps N (a) and entropy threshold γ (b).

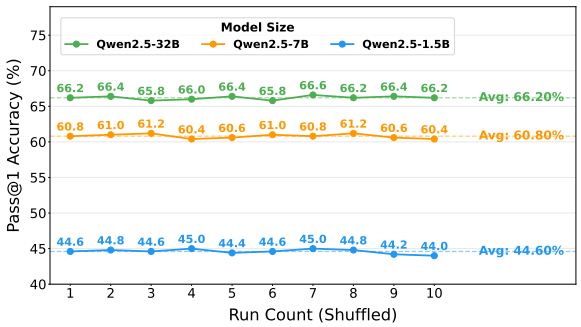

Figure 3: Performance stability of SEER across 10 runs with randomly shuffled execution orders on MATH500.

performance drop. This underscores the critical role of the gating mechanism: by selectively activating retrieval only during high-uncertainty moments, ensuring that the model receives necessary guidance without disrupting the reasoning flow.

	MATH500 Accuracy (%)			
Backbone	**SEER**	**w/o MCMC**	**w/o Distill**	**w/o Gate**
Qwen2.5-1.5B	44.6	39.4	39.6	40.4
Qwen2.5-7B	60.8	58.0	56.0	56.6
Qwen3-8B	67.4	63.2	62.0	62.8
Qwen3-14B	69.6	65.8	64.6	65.2
Qwen2.5-32B	66.2	63.8	64.2	63.4

Table 4: Ablation study on MATH500 across different size of models.

4.4 Parameter Sensitivity Analysis

We evaluate the sensitivity of our approach to two pivotal hyperparameters: the number of MCMC steps (N) and the entropy threshold (γ) across five model scales. As depicted in Figure 2(a), the accuracy on MATH500 consistently improves as N increases from 5 to 25. We observe a notable performance surge before $N = 15$ followed by a gradual plateau, suggesting that $N = 15$ serves as an efficient elbow point for balancing computation and reasoning quality. Simultaneously, under identical decoding configurations (e.g., a fixed temperature), Figure 2(b) reveals a performance curve regarding the entropy threshold γ, where all model scales achieve their peak accuracy at $\gamma = 0.6$. The performance degradation observed at extreme threshold values, such as $\gamma = 0.2$ or $\gamma = 1.0$, highlights the necessity of a moderate gating threshold to filter reasoning paths effectively.

5 Conclusion

In conclusion, we propose SEER, a fully unsupervised framework for self-evolving reasoning in large language models. SEER replaces external supervision with model-intrinsic verification, identifying inputs with emergent learning potential and validating experiences through consistency gains and experience confidence. By integrating these intrinsic signals with MCMC-based exploration and teacher-guided distillation, SEER constructs a high-quality experience memory bank that enables autonomous self-evolution without access to ground-truth labels. Extensive experiments demonstrate that SEER consistently outperforms strong baselines on complex reasoning tasks, highlighting its effectiveness as a training-free approach to self-evolving reasoning.

Limitations

To further enhance the framework's robustness and generalization, future work will focus on theoretically characterizing the relationship between sampling diversity and experience convergence. Additionally, we aim to integrate large-scale unsupervised experience accumulation to systematically support reasoning across broader domains.

References

Emre Can Acikgoz, Cheng Qian, Heng Ji, Dilek Hakkani-Tür, and Gokhan Tur. 2025. Self-improving LLM agents at test-time. *CoRR*, abs/2510.07841.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and 1 others. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, and 39 others. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Daixuan Cheng, Shaohan Huang, Xuekai Zhu, Bo Dai, Wayne Xin Zhao, Zhenliang Zhang, and Furu Wei. 2025. Reasoning with exploration: An entropy perspective. *arXiv preprint arXiv:2506.14758*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Andre Wang He, Daniel Fried, and Sean Welleck. 2025a. Rewarding the unlikely: Lifting grpo beyond distribution sharpening. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 25559–25571.

Kaiwen He, Zhiwei Wang, Chenyi Zhuang, and Jinjie Gu. 2025b. Recon-act: A self-evolving multi-agent browser-use system via web reconnaissance, tool generation, and task execution. *CoRR*, abs/2509.21072.

Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Aayush Karan and Yilun Du. 2025. Reasoning with sampling: Your base model is smarter than you think. *Preprint*, arXiv:2510.14901.

Sunbowen Lee, Qingyu Yin, Chak Tou Leong, Jialiang Zhang, Yicheng Gong, Shiwen Ni, Min Yang, and Xiaoyu Shen. 2025. Probing the difficulty perception mechanism of large language models. *arXiv preprint arXiv:2510.05969*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, and 1 others. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Siheng Li, Zhanhui Zhou, Wai Lam, Chao Yang, and Chaochao Lu. 2025a. Repo: Replay-enhanced policy optimization. *arXiv preprint arXiv:2506.09340*.

Zhiyu Li, Shichao Song, Chenyang Xi, Hanyu Wang, Chen Tang, Simin Niu, Ding Chen, Jiawei Yang, Chunyu Li, Qingchen Yu, and 1 others. 2025b. Memos: A memory os for ai system. *arXiv preprint arXiv:2507.03724*.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *Preprint*, arXiv:2305.20050.

Zihan Liu, Shun Zheng, Xumeng Wen, Yang Wang, Jiang Bian, and Mao Yang. 2025. Deep self-evolving reasoning. *Preprint*, arXiv:2510.17498.

Siru Ouyang, Jun Yan, I Hsu, Yanfei Chen, Ke Jiang, Zifeng Wang, Rujun Han, Long T Le, Samira Daruki, Xiangru Tang, and 1 others. 2025. Reasoningbank: Scaling agent self-evolving with reasoning memory. *arXiv preprint arXiv:2509.25140*.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, and 1 others. 2024. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. 2023. GPQA: A graduate-level google-proof q&a benchmark. *CoRR*, abs/2311.12022.

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Preprint*, arXiv:2303.11366.

Shaobo Wang, Zhengbo Jiao, Zifan Zhang, Yilang Peng, Xu Ze, Boyu Yang, Wei Wang, Hu Wei, and Linfeng Zhang. 2025a. Socratic-zero : Bootstrapping reasoning via data-free agent co-evolution. *CoRR*, abs/2509.24726.

Shenzhi Wang, Le Yu, Chang Gao, Chujie Zheng, Shixuan Liu, Rui Lu, Kai Dang, Xionghui Chen, Jianxin Yang, Zhenru Zhang, and 1 others. 2025b. Beyond the 80/20 rule: High-entropy minority tokens drive effective reinforcement learning for llm reasoning. *arXiv preprint arXiv:2506.01939*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and Botian Shi. 2025a. Evolver: Self-evolving LLM agents through an experience-driven lifecycle. *CoRR*, abs/2510.16079.

Rong Wu, Xiaoman Wang, Jianbiao Mei, Pinlong Cai, Daocheng Fu, Cheng Yang, Licheng Wen, Xuemeng Yang, Yufan Shen, Yuxin Wang, and 1 others. 2025b. Evolver: Self-evolving llm agents through an experience-driven lifecycle. *arXiv preprint arXiv:2510.16079*.

Yuhuai Wu, Markus N Rabe, DeLesley Hutchins, and Christian Szegedy. 2022. Memorizing transformers. *arXiv preprint arXiv:2203.08913*.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Kristian Kersting, Jeff Z Pan, Hinrich Schütze, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Yang Yue, Zhiqi Chen, Rui Lu, Andrew Zhao, Zhaokai Wang, Shiji Song, and Gao Huang. 2025. Does reinforcement learning really incentivize reasoning capacity in llms beyond the base model? *arXiv preprint arXiv:2504.13837*.

Runzhe Zhan, Yafu Li, Zhi Wang, Xiaoye Qu, Dongrui Liu, Jing Shao, Derek F Wong, and Yu Cheng. 2025. Exgrpo: Learning to reason from experience. *arXiv preprint arXiv:2510.02245*.

Yifan Zhang and Team Math-AI. 2024. American invitational mathematics examination (aime) 2024.

Yifan Zhang and Team Math-AI. 2025. American invitational mathematics examination (aime) 2025.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025. Absolute zero: Reinforced self-play reasoning with zero data. *Preprint*, arXiv:2505.03335.

Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. 2024. Language agent tree search unifies reasoning acting and planning in language models. *Preprint*, arXiv:2310.04406.

A Details on Power Sampling Strategy

In this section, we provide a theoretical analysis comparing our proposed approach with standard temperature sampling following previous work (Karan and Du, 2025), followed by the complete pseudo-code for the MCMC implementation.

A.1 Comparison with Temperature Sampling

A distinction must be drawn between standard temperature sampling (with temperature $\tau = 1/\alpha$) and sampling from the globally sharpened power distribution $P_\theta(z)^\alpha$. While both methods serve to concentrate probability mass on high-likelihood regions, they differ fundamentally in the commutativity of aggregation and exponentiation operations regarding future probabilities.

A.1.1 Mathematical Formulation

Let z_t denote the token at step t, and let $z_{>t}$ represent a complete future trajectory.

Standard Temperature Sampling: Standard sampling operates *locally*. At step t, the model first computes the marginal probability of z_t by marginalizing over all futures, and subsequently sharpens this sum.

$$P_{\text{temp}}(z_t|z_{<t}) \propto (P_\theta(z_t|z_{<t}))^\alpha = \left(\sum_{z_{>t}} P_\theta(z_t, z_{>t}|z_{<t})\right)^\alpha \quad (5)$$

Power Distribution Sampling: In contrast, the Power Distribution is defined on the *global* sequence space. The formulation necessitates sharpening the joint probability of every complete path prior to marginalization.

$$P_{\text{power}}(z_t|z_{<t}) \propto \sum_{z_{>t}} (P_\theta(z_t, z_{>t}|z_{<t}))^\alpha = \sum_{z_{>t}} P_\theta(z_t, z_{>t}|z_{<t})^\alpha \quad (6)$$

A.1.2 Divergence due to Jensen's Inequality

The mathematical discrepancy between Eq. 5 and Eq. 6 induces a critical behavioral divergence derived from Jensen's Inequality. Since the function $f(x) = x^\alpha$ is convex for $\alpha > 1$, the operations of summation and exponentiation do not commute, leading to distinct sensitivities:

Algorithm 1 Diversity-Enhanced Exploration via MCMC (Power Sampling)

Require: Query x, Base Model P_θ, Sharpening α, Steps N, Burn-in B
Ensure: Set of diverse trajectories $\mathcal{T}$

Define target density: $\pi(z) \propto P_\theta(z|x)^\alpha$
Initialize $z_0 \sim P_\theta(\cdot|x)$; $\mathcal{T} \leftarrow \emptyset$
for $t = 0$ **to** $N-1$ **do**
  Step 1: Proposal (Local Edit)
    Sample cut-off index $k \sim \text{Uniform}(1, |z_t|)$.
    Keep prefix $z_{<k}$ and sample suffix $z'_{>k} \sim P_\theta(\cdot \mid x, z_{<k})$.
    Form candidate $z' = [z_{<k}; z'_{>k}]$.
  Step 2: Acceptance Decision
    Compute acceptance probability A:

$$A = \min\left(1, \underbrace{\frac{\pi(z')}{\pi(z_t)}}_{\text{Target Ratio}} \cdot \underbrace{\frac{P_\theta(z_{t,>k}|z_{<k})}{P_\theta(z'_{>k}|z_{<k})}}_{\text{Correction } q(z_t|z')/q(z'|z_t)}\right)$$

  Step 3: Update and Collect
    Sample $u \sim \mathcal{U}[0,1]$.
  if $u < A$ **then**
    $z_{t+1} \leftarrow z'$
  else
    $z_{t+1} \leftarrow z_t$
  end if
  {Collect samples after burn-in period}
  if $t \geq B$ **then**
    $\mathcal{T} \leftarrow \mathcal{T} \cup \{z_{t+1}\}$
  end if
end for

- **Sensitivity to Marginal Aggregate (Standard Temperature Sampling):** By summing prior to exponentiation, standard sampling preserves the rank ordering of tokens based on their total marginal mass. It favors high-entropy states where the probability mass is distributed across many moderately likely futures, rather than identifying sparse, high-likelihood trajectories.

- **Sensitivity to Pathwise Maxima (Power Sampling):** By exponentiating prior to summation, the Power Distribution disproportionately amplifies tokens associated with specific high-likelihood complete trajectories. The convexity of the power function suppresses the contribution of low-probability paths, effectively acting as a soft-maximization over future outcomes.

This property renders Power Sampling theoretically superior for reasoning tasks involving intermediate steps that possess low marginal probability but are necessary precursors to the global max-

imum likelihood trajectory.

A.1.3 Numerical Analysis

We illustrate this divergence through a modified numerical analysis with $\alpha = 2$ (temperature $\tau = 0.5$). Consider a decision boundary at step t between two candidate tokens, $z^{(A)}$ and $z^{(B)}$.

1. **Candidate A (High-Entropy):** Leads to 3 distinct diverse futures, each with moderate probability $p = 0.15$. Total marginal probability: $\sum p = 0.45$.

2. **Candidate B (Low-Entropy):** Leads to a single distinct future with probability $p = 0.40$ (assuming negligible mass elsewhere). Total marginal probability: 0.40.

Under the base model distribution ($\alpha = 1$), Candidate A is strictly preferred because its aggregated mass is higher ($0.45 > 0.40$).

Scenario 1: Standard Temperature Sampling ($\alpha = 2$) The standard mechanism sharpens the *aggregated* marginals first:

- Weight for A: $(0.15 + 0.15 + 0.15)^2 = 0.45^2 = \mathbf{0.2025}$
- Weight for B: $0.40^2 = \mathbf{0.1600}$

Result: The model **retains the preference for A** ($0.2025 > 0.1600$). The sampling process fails to penalize Candidate A for accumulating its mass from multiple mediocre outcomes.

Scenario 2: Power Distribution Sampling ($\alpha = 2$) The power distribution sums the *sharpened* path likelihoods:

- Weight for A: $0.15^2 + 0.15^2 + 0.15^2 = 3 \times 0.0225 = \mathbf{0.0675}$
- Weight for B: $0.40^2 = \mathbf{0.1600}$

Result: The preference is **strongly reversed in favor of B**. Since $0.1600 \gg 0.0675$, the Power Distribution correctly identifies that Candidate B enables the global maximum likelihood trajectory. This highlights the method's ability to filter out "noisy" high-entropy tokens in favor of "sparse" pivotal tokens.

A.2 Algorithmic Workflow of Power Sampling

In this section, we detail the procedural workflow of our MCMC-based Power Sampling. The core objective is to simulate samples from the target distribution $\pi(z|x) \propto P_\theta(z|x)^\alpha$ by constructing a Markov chain. As outlined in Algorithm 1, the process begins with an initial trajectory generated via standard sampling. In each iteration, we perturb the current state to propose a new candidate reasoning path and apply the Metropolis-Hastings acceptance criterion. This mechanism ensures that the stationary distribution of the chain aligns with our target Power Distribution, effectively guiding the model toward high-likelihood reasoning modes.

B Calculation of Consistency Metrics

In this section, we detail the mathematical formulation of the SC score and entropy used in our filtering mechanism. These metrics are computed in a label-free manner, relying solely on the distribution of sampled reasoning paths.

B.1 Self-consistency metric

Sampling and Notation. For a given input problem x, we sample K reasoning paths $\{r_1, r_2, \ldots, r_K\}$ from the policy model π_θ. Let y_i denote the final answer derived from the path r_i. As ground truth labels are not utilized during this phase, we estimate the "correct" answer based on the majority consensus.

Majority Vote. We first identify the majority consensus answer $\hat{y}$, which is the answer with the highest frequency among the sampled paths:

$$\hat{y} = \arg\max_{v \in \mathcal{V}} \sum_{i=1}^{K} \mathbb{I}(y_i = v) \tag{7}$$

where $\mathcal{V}$ represents the set of unique answers generated across the K paths, and $\mathbb{I}(\cdot)$ is the indicator function which equals 1 if the condition is met and 0 otherwise.

Consistency Score (S). The consistency score quantifies the model's confidence in the consensus answer. It is defined as the empirical probability of observing $\hat{y}$:

$$S(x) = \frac{1}{K} \sum_{i=1}^{K} \mathbb{I}(y_i = \hat{y}) \tag{8}$$

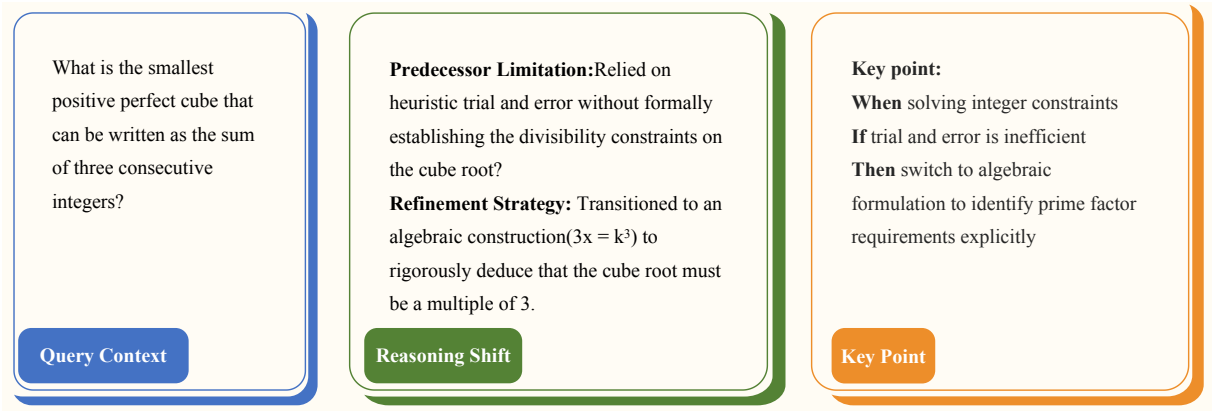

Figure 4: Visualization of a standardized entry in the SEER memory bank.

A score $S(x) \to 1$ indicates that the model consistently converges to a single solution (low uncertainty), whereas $S(x) \to 0$ implies high divergence among reasoning paths.

B.2 Average Process Entropy

To capture the model's intrinsic uncertainty throughout the entire reasoning process (rather than just the final answer distribution), we calculate the average token-level entropy.

Let each sampled reasoning path r_i consist of a sequence of L_i tokens, denoted as $\{w_{i,1}, w_{i,2}, \ldots, w_{i,L_i}\}$. At each time step t, the model outputs a probability distribution over the vocabulary $\mathcal{V}$, denoted as $P(w \mid x, w_{i,<t})$. The entropy at this specific step is calculated as:

$$H_{i,t} = -\sum_{w \in \mathcal{V}} P(w \mid x, w_{i,<t}) \log P(w \mid x, w_{i,<t}). \quad (9)$$

The final entropy metric E (e.g., E_1) is defined as the average entropy across all steps and all K sampled paths:

$$E(x) = \frac{1}{K} \sum_{i=1}^{K} \left(\frac{1}{L_i} \sum_{t=1}^{L_i} H_{i,t} \right). \quad (10)$$

A lower value of E indicates that the model is more confident and decisive in its step-by-step reasoning process, while a higher value suggests high uncertainty or confusion during generation.

C Uncertainty Quantification Details

In this section, we provide the detailed formulation for the uncertainty quantification mechanism described in Section 3.3.

Raw Token Entropy. At any generation step t, given the input query x and the preceding context $z_{<t}$, the model outputs a probability distribution $P_\theta(v \mid x, z_{<t})$ over the vocabulary $\mathcal{V}$. Similar to B.2, the raw H_t is calculated as :

$$H_t = -\sum_{v \in \mathcal{V}} P_\theta(v \mid x, z_{<t}) \log P_\theta(v \mid x, z_{<t}) \quad (11)$$

Smoothed Entropy via Sliding Window. We observe that raw entropy H_t is susceptible to high-frequency noise caused by trivial lexical variances (e.g., the choice between interchangeable function words or punctuation) that do not necessarily reflect genuine reasoning difficulties. To filter out these token-level fluctuations and capture the cognitive load over a logical segment, we employ a sliding window smoothing mechanism.

Let w be the window size (set to $w = 5$ in our experiments). The Smoothed Shannon Entropy $\bar{H}_t$ is calculated as the moving average of the raw entropies over the local context window:

$$\bar{H}_t = \frac{1}{\min(t, w)} \sum_{j=0}^{\min(t,w)-1} H_{t-j}. \quad (12)$$

This smoothing ensures that the injection gate is triggered only by sustained uncertainty, indicating a potential reasoning gaprather than transient spikes.

Gating Mechanism. The adaptive injection gate $\mathbb{G}_t \in \{0, 1\}$ operates as a binary switch based on the smoothed metric. For a calibrated threshold γ, the decision experience is formalized as:

$$\mathbb{G}_t = \mathbb{I}(\bar{H}_t > \gamma) = \begin{cases} 1, & \text{if } \bar{H}_t > \gamma \\ 0, & \text{otherwise} \end{cases}. \quad (13)$$

where $\mathbb{I}(\cdot)$ is the indicator function. TThis ensures that external retrieval is activated only when the

model's internal confidence falls below the threshold.

D Memory Entry Schema

To strictly align with the *Experience Memory Construction* process defined in Section 3.2, each memory entry $e_i \in \mathcal{B}$ is stored as a streamlined key-value structure. The schema focuses exclusively on the core components of the distillation pipeline:

- **`query_context`** (q_i): The original problem state that triggered the reasoning process.
- **`reasoning_shift`** (Δ): The contrastive rationale extracted by the teacher model. Instead of merely correcting errors, it explicitly maps the evolutionary divergence between the predecessor trajectory ($z^{(t-1)}$) and its refined successor ($z^{(t)}$). This component captures the logical optimization stepsuch as moving from heuristic guessing to rigorous derivation, or from ambiguous description to precise definition.
- **`key_point`** (k_i): The final distilled heuristic. This field represents the generalized experience, verified for high information density and reusability across different problem instances.

Figure 4 visualizes a concrete example of such an entry, demonstrating how the abstract mathematical definitions translate into a structured data format.

E Case Study

To intuitively understand the working mechanism of SEER, we present a concrete case study in Figure 5.

Case Analysis. As shown in Figure 5, the model is tasked with a number theory problem: *"What is the least positive integer multiple of 30 that can be written with only the digits 0 and 2?"*

- **Native Failure :** The native model (without SEER) attempts to solve the problem but fails to strictly adhere to the "least" constraint or correctly apply divisibility rules, outputting an incorrect answer.
- **Experience Retrieval & Injection:** Triggered by the high entropy, SEER halts the current generation and select, rerank and aggregate the experience that is *"Consider the properties of multiples of 30."*. This concise experience acts as a navigational guide for the subsequent inference.
- **SEER Refinement:** Equipped with the injected experience, SEER regenerates the reasoning path. As observed in the "Regenerate" phase, the model now explicitly articulates the divisibility rule: *"A number is divisible by 30 if and only if it is divisible by both 3 and 10."* Following this structured logic, it correctly identifies that the sum of digits must be divisible by 3, leading to the correct least integer, $\mathbf{2220}$.

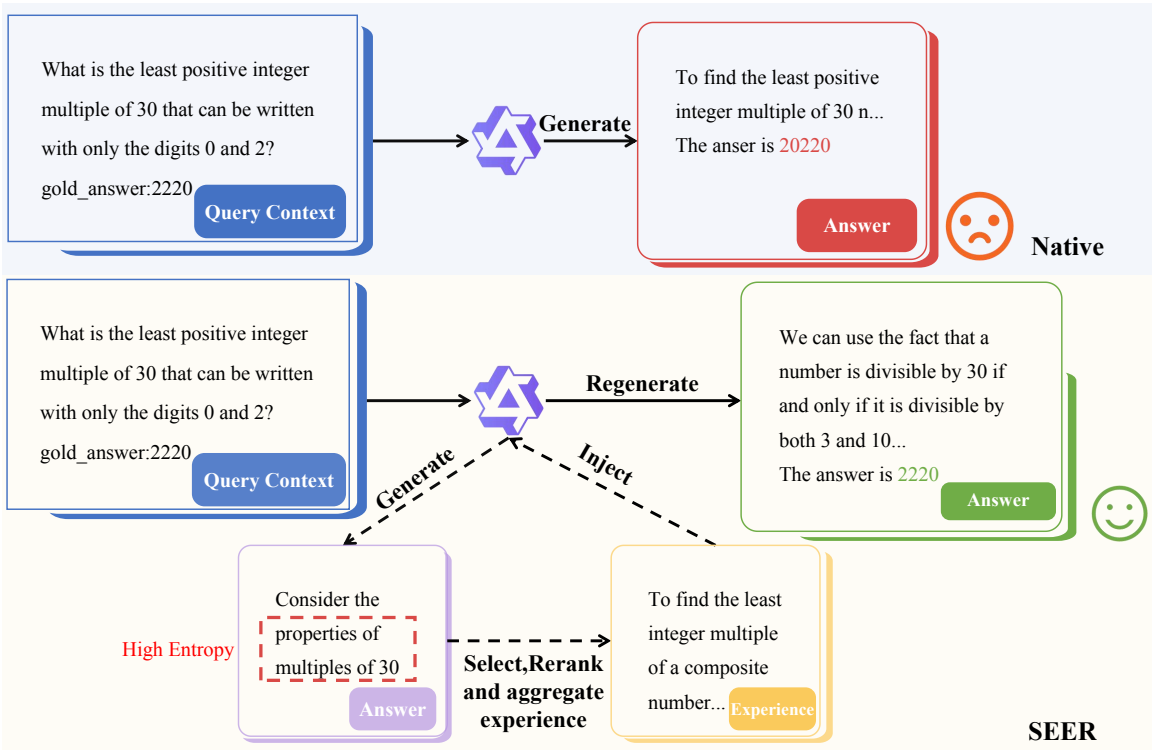

Figure 5: **Mechanism Visualization.** Comparison between the native model and SEER on a number theory problem. When the native model produces a high-entropy response, SEER triggers the memory bank to select, rerank and aggregate experience, enabling the model to regenerate a correct reasoning path.