NATIVE NON-MONOTONIC AUTOREGRESSIVE SEQUENCE MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive models generate sequences monotonically, *i.e.*, every sampled token, even if erroneous, becomes a permanent condition for all subsequent steps, and the cost grows with the length of the generation. Non-monotonic autoregressive models address this limitation with a learned erase token, while existing designs roll back one token per command and learn from task-specific augmented traces, so the capability neither scales to long-form generation nor transfers beyond the tasks that supervise it. To this end, we propose **N^2-MARS**, **N**ative **N**on-**M**onotonic **A**uto**R**egressive **S**equence models, which introduce a rollback command $\langle$UNDO$\rangle N \langle$/UNDO$\rangle$ for autoregressive sequence models to roll back N tokens as a native generation primitive learned from open-domain text. In particular, we construct a 9.5B on-policy rollback corpus by using model completions of open-domain texts and further employ LLM-as-Judge to annotate the completions, forming the rollback trajectories that allow models to recognize and roll back their own errors. Afterward, we design hybrid continued pre-training (HyCPT), a single stage that unifies continued pre-training and masked supervised fine-tuning under source-specific loss masks, followed by reinforcement learning with outcome rewards that refines when the command fires. Extensive experiments on three long-context generation benchmarks and nine general benchmarks, across four backbones from 3B to 9B parameters in two model families, demonstrate N^2-MARS's efficiency and effectiveness. Our source code, corpus, and checkpoints will be publicly released upon acceptance.

1 INTRODUCTION

Autoregressive sequence models generate tokens monotonically, where each sampled token is appended to an immutable sequence and conditions every later step [2, 55]. This monotonic behavior makes generation susceptible to error propagation, *i.e.*, a single erroneous token corrupts the context that all subsequent tokens depend on, and the model has no mechanism to withdraw it [77, 15]. The cost of this commitment grows with output length; *i.e.*, in long-form generation such as reports and programs [73, 45, 67, 4], an early mistake sits far from the generation frontier and becomes increasingly expensive to repair by starting over.

To mitigate error propagation, post-hoc methods [40, 52, 13, 33] judge a finished draft and regenerate it, which requires an additional generation pass. Ad-hoc methods interleave revision with decoding through special tokens that reset a response [78, 50], revert a reasoning step [71], or erase the previous token [12, 39]. Specifically, N-MARS [39] demonstrates that a learned erase token, trained with masked supervised fine-tuning (mSFT), allows a model to correct itself within a single forward pass. However, existing methods still face the following limitations: (i) *Atomic rollback.* Each command removes a single token or a fixed unit, so the number of commands grows linearly with the distance to the error. (ii) *Task-bound supervision.* Error-correction traces are built from the references of a target benchmark, so the behavior is tied to the supervising task rather than being a native capability of the model. (iii) *Zero-lag detection.* Training places the command exactly at the point of divergence, whereas errors in open-ended text typically become evident only after the model has continued past them. This naturally raises a research question: *Can rollback be installed into a language model as a native generation capability, learned from its own mistakes on open-domain text, so that it transfers to long-form generation at scale?* To this end, we propose **N^2-MARS**, **N**ative

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.

(a) Verbal self-correction

...the report covers fiscal year 2019, when the ... Wait, that was wrong. The report covers 2021, when the agency ...

(b) N-MARS

...the report covers fiscal year ~~2019, when the agency ...~~ ⟨UNDO⟩ ⟨UNDO⟩ ⟨UNDO⟩ ⋯⟨UNDO⟩ 2021, when the agency ...

(c) N^2-MARS

...the report covers fiscal year ~~2019, when the agency first reported the ...~~ <UNDO>N</UNDO> 2021, when the agency ...

Figure 1: Native rollback for long-form generation. Three revision mechanisms for the same N error tokens with repair $\mathbf{y}$, where N^2-MARS rolls back the whole span through a single command $\langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle$.

Non-**M**onotonic **A**uto**R**egressive **S**equence models, a framework that teaches pretrained language models a parameterized rollback command $\langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle$, which deletes the last N visible tokens and continues with a repaired suffix in the same pass, as illustrated in Figure 1. To handle the first limitation, the command carries its span as an argument spelled with the existing digit tokens, so a correction of any length costs a single command and only two new vocabulary entries. To handle the second limitation, we construct an on-policy rollback corpus from open-domain text, *i.e.*, we sample completions of web, code, mathematics, book, and report documents from the model itself, label them with a judge, and convert the rejected ones into rollback trajectories and the accepted ones into paired clean completions that teach the model when not to roll back. To handle the third limitation, we place the command at the end of the rejected completion rather than at the point of divergence, so the model learns to detect an error from the evidence that accumulates after it. Afterward, we design hybrid continued pre-training (HyCPT), a single stage that unifies continued pre-training and mSFT in one mixture with source-specific loss masks, and refine the trigger policy through reinforcement learning with outcome rewards. Our contributions can be summarized as follows:

- **Native Non-Monotonic Sequence Modeling.** We propose N^2-MARS, which installs a parameterized rollback command into pretrained language models as a general generation primitive. To the best of our knowledge, this is the first work that learns multi-token, single-pass rollback of visible tokens from open-domain text.
- **On-Policy Rollback Corpus and Hybrid Continued Pre-Training.** We introduce a judged on-policy rollback corpus sampled from the model's own completions across seven open-domain sources, and HyCPT, a training stage with source-specific masking that teaches rollback and repair without reinforcing the model's own errors.
- **Long-Context and General Benchmark Evaluation.** We conduct comprehensive experiments on three long-context generation benchmarks and nine general benchmarks across four backbones from 3B to 9B parameters, demonstrating the performance across the evaluated tasks and backbones of N^2-MARS.

2 Methodology

In this section, we present N^2-MARS, whose overall framework is illustrated in Figure 2.

We formulate autoregressive sequence modeling as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{P}, r, \gamma)$, where a state $\mathbf{s}_t = (x_1, \ldots, x_t)$ from state space $\mathcal{S}$ is a sequence of tokens from the vocabulary $\mathcal{V}$, action space $\mathcal{A}$ is a finite set of the vocabulary $\mathcal{V}$, and a policy p_θ parameterized by θ is a distribution over the next action x_{t+1} from the action space $\mathcal{A}$ given the current state $\mathbf{s}_t$. In N^2-MARS, we extend the action space with two marker tokens, *i.e.*, $\mathcal{A} = \mathcal{V} \cup \{\langle\text{UNDO}\rangle, \langle/\text{UNDO}\rangle\}$, and define a *rollback command* that rolls back the last N tokens in the current sequence as the token sequence

$$\langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle = \big(\langle\text{UNDO}\rangle,\, d_1, \ldots, d_m,\, \langle/\text{UNDO}\rangle\big), \tag{1}$$

where $d_1 \cdots d_m$ is the decimal spelling of N and $d_i \in \mathcal{V}$.

Efficient Non-Monotonic Generation. The model decodes a *raw trajectory* τ over $\mathcal{A}$ with a standard autoregressive decoder, and a deterministic interpreter Φ reduces it to the *visible sequence* $\mathbf{s} = \Phi(\tau)$ without any auxiliary model or additional pass. For a trajectory τ followed by an ordinary token $x \in \mathcal{V}$ or by a command $\langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle$, the interpreter is defined recursively as

$$\Phi(\tau \| x) = \Phi(\tau) \| x, \qquad \Phi\big(\tau \| \langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle\big) = \Phi(\tau)_{1:|\Phi(\tau)|-N}, \tag{2}$$

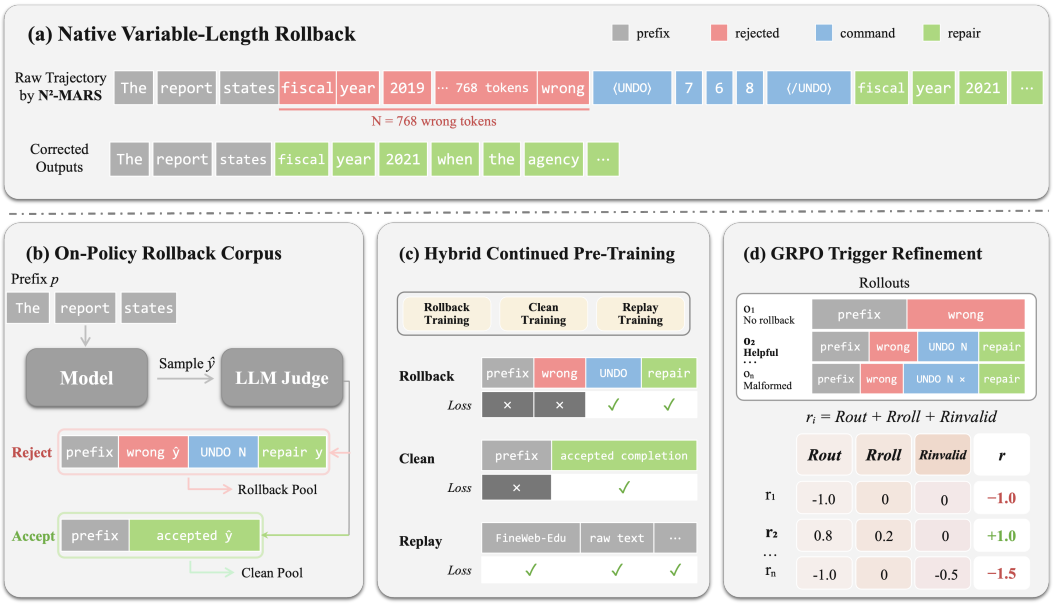

Figure 2: The overall framework of **N^2-MARS**.

with $\|$ denoting concatenation.

2.1 Open-Domain On-Policy Rollback Corpus Construction

To learn when to roll back, a model has to see three things in one training sequence, *i.e.*, a mistake, the command that removes it, and the corrected text that follows [72]. The mistakes should be the model's own errors, since training on synthetically injected errors teaches the injection pattern rather than self-correction [30, 64]. Open-domain documents provide both at once, *i.e.*, we let the model continue a real document, so the mistakes are its own, and the document's original text serves as the repair.

On-Policy Sampling. Given a document from a source corpus, we split its token sequence $\mathbf{x}$ into a prefix and reference sequence, denoted as $\mathbf{p}$ and $\mathbf{y}$, such that $\mathbf{x} = \mathbf{p}\|\mathbf{y}$. Then we sample a completion $\hat{\mathbf{y}} \sim p_{\theta_0}(\cdot \mid \mathbf{p})$ from the base model. The completion is cut at the last sentence boundary for prose or a line boundary for code, so that $\hat{\mathbf{y}}$ ends at a complete unit. Then we obtain a triplet $(\mathbf{p}, \hat{\mathbf{y}}, \mathbf{y})$ for the following rollback trajectory construction.

Judging. Simply constructing the training trajectory from the triple $(\mathbf{p}, \hat{\mathbf{y}}, \mathbf{y})$ is insufficient to establish an error, as a different completion may still be correct. Therefore, we adopt a judge model J to read the prefix sequence $\mathbf{p}$, sampled completion $\hat{\mathbf{y}}$, and reference sequence $\mathbf{y}$, and annotate whether $\hat{\mathbf{y}}$ is consistent and correct with the prefix and internally plausible. As a result, we obtain a rollback pool $\mathcal{D}_{\text{undo}}$ that contains the wrong triplet and a clean pool $\mathcal{D}_{\text{clean}}$ for acceptable triplets. The judge prompt is provided in Appendix B.3.

Rollback Trajectory Construction. For a sample in $\mathcal{D}_{\text{undo}}$, we place a rollback command at the end of the rejected completion, following with the reference:

$$\mathbf{z} = \mathbf{p} \,\|\, \hat{\mathbf{y}} \,\|\, \langle \text{UNDO} \rangle N \langle /\text{UNDO} \rangle \,\|\, \mathbf{y}, \qquad N = |\hat{\mathbf{y}}|, \tag{3}$$

so that the command rolls back exactly the text that the judge rejected. Figure 3 traces one sample of $\mathcal{D}_{\text{undo}}$ through this construction, where the rejected completion becomes the rolled-back span, and one $\langle \text{UNDO} \rangle 135 \langle /\text{UNDO} \rangle$ makes the interpreter return the prefix followed by the reference.

Paired Clean Completions. A model trained only on rollback trajectories learns to roll back indiscriminately, so the mixture pairs them with clean completions from $\mathcal{D}_{\text{clean}}$ that carry no command, whose lengths are drawn from the observed distribution of rollback spans, so that the completion length is only a weak cue for whether to rollback.

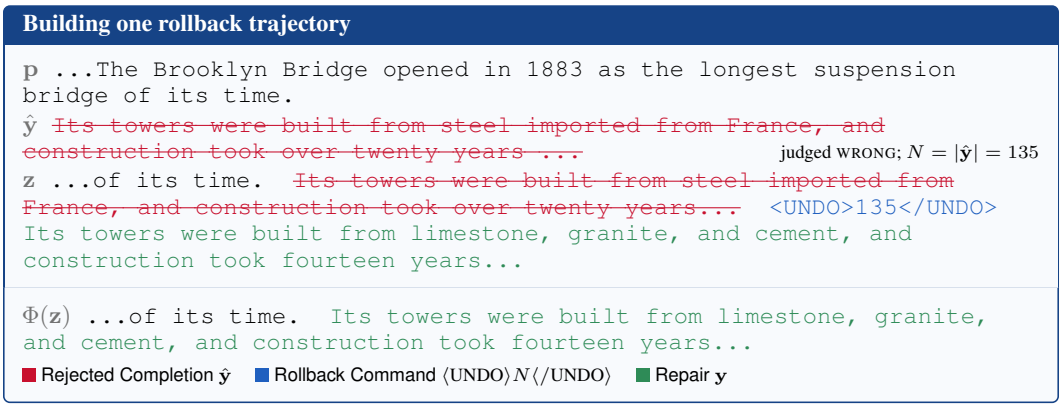

Figure 3: Building one rollback trajectory. A completion rejected by the judge becomes the rolled-back span, a single command $\langle\text{UNDO}\rangle N\langle/\text{UNDO}\rangle$ with $N = |\hat{\mathbf{y}}|$ follows it, and the reference serves as the repair, so the interpreter returns the prefix followed by the reference.

2.2 Hybrid Continued Pre-Training with Source-Specific Masking

A natural recipe would first integrate the command syntax through continued pre-training and then teach rollback through mSFT. However, continued pre-training with full labels degrades the backbone's general capability [28], and its protection comes from the replay mixture rather than the full-label objective. Inspired by this, we merge the two stages into *hybrid continued pre-training* (HyCPT), a single stage whose labeling policy differs per source, as illustrated in Figure 2(c) and detailed in Table 8 (Appendix B). Formally, let $\mathcal{S} = \{\text{raw}, \text{inst}, \text{undo}, \text{clean}\}$ denote the sources with mixture weights π_s, and let $M^{(s)} \in \{0,1\}^T$ be the source-specific mask of a sequence $\mathbf{z} = (z_1, \ldots, z_T)$. The HyCPT objective is

$$\mathcal{L}_{\text{HyCPT}}(\theta) = -\sum_{s\in\mathcal{S}} \pi_s \, \mathbb{E}_{\mathbf{z}\sim\mathcal{D}_s}\Big[\sum_{t=1}^{T} M_t^{(s)} \log p_\theta(z_t \mid z_{<t})\Big]. \tag{4}$$

For rollback trajectories, $M_t = 0$ for every position in $\mathbf{p}\|\hat{\mathbf{y}}$, so the model conditions on its erroneous history without being trained to reproduce it, and $M_t = 1$ for the command, the repair, and the end-of-sequence token.

2.3 Trigger Refinement with Reinforcement Learning

HyCPT teaches the model how to roll back and repair, whereas whether a rollback pays off depends on the outcome of the whole generation. So we refine the policy with Group Relative Policy Optimization (GRPO) [51] on the collapsed output. For a prompt $\mathbf{x}$, we sample a group of G raw trajectories $\{\tau_i\}_{i=1}^{G}$, interpret each into $\Phi(\tau_i)$, and obtain a verdict $v(\tau_i) \in \{0,1\}$ from a rule-based verifier on verifiable prompts and from the judge J on open-domain prompts. The reward is

$$\mathcal{R}(\tau) = \underbrace{r_{v(\tau),\,c(\tau)}}_{\text{outcome}} + \underbrace{\beta_{\text{help}}\,\mathbb{1}[\text{helped}(\tau)]}_{\text{effective rollback}} - \underbrace{\lambda_{\text{mal}}\,\mathbb{1}[\text{malformed}(\tau)]}_{\text{invalid command}}, \tag{5}$$

where $c(\tau)$ indicates whether τ contains a valid command that rolled back a wrong visible answer, $r_{v,c}$ is an outcome matrix that ranks a correct output above an incorrect one for every c, and $\text{helped}(\tau) = 1$ if a visible prefix immediately before a valid command is wrong while the final output is correct. The bonus is paid only for a verified wrong-to-right transition, a valid command on an output that was already correct earns nothing beyond the outcome term, and $r_{0,1} > r_{0,0}$ keeps a rollback of a wrong answer from being pruned on prompts where no sampled trajectory succeeds. The objective follows Shao et al. [51] with a KL penalty toward the HyCPT policy, and the reward constants are listed in Appendix A.3.

Table 1: Performance on general benchmarks across four backbones from two model families. Within each backbone, **bold** is the best and underline the second best per column; **Rank** is the mean rank of the method over the nine benchmarks within its backbone, ties sharing the mean rank.

Backbone	Method	General			Code		Math		Planning		Rank ↓
		MMLU	MMLU-PRO	IFEVAL	HUMANEVAL	MBPP	GSM8K	MATH	COUNT	SUDOKU	
Llama-3.2-3B	Base	55.0	25.0	12.0	26.8	39.4	28.4	6.4	0.0	0.0	3.7
	w/ CPT	54.6	23.9	16.8	28.6	42.4	33.9	10.6	0.0	0.9	3.2
	w/ HyCPT	56.1	28.2	20.5	30.5	41.8	40.2	11.5	1.9	2.1	2.1
	N^2-MARS	**56.7**	**28.7**	**22.4**	**33.7**	**46.0**	**48.4**	**14.3**	**2.4**	**3.5**	**1.0**
Llama-3.1-8B	Base	64.1	35.8	14.1	36.6	49.2	53.2	20.0	0.0	0.0	3.6
	w/ CPT	63.0	35.6	18.9	39.1	50.9	55.6	24.8	0.0	1.2	3.2
	w/ HyCPT	63.7	37.0	20.8	40.5	50.6	60.8	26.1	2.8	2.6	2.2
	N^2-MARS	**64.4**	**37.8**	**27.7**	**42.7**	**54.8**	**70.7**	**27.8**	**3.2**	**4.9**	**1.0**
Qwen3.5-4B	Base	73.2	52.6	52.3	50.0	52.8	81.8	50.9	2.0	0.0	3.6
	w/ CPT	73.8	51.2	53.3	55.2	53.4	80.8	55.2	4.1	0.0	3.0
	w/ HyCPT	**74.1**	53.7	52.6	59.1	57.4	83.6	60.9	6.1	0.0	2.1
	N^2-MARS	72.9	**55.8**	**55.7**	**59.6**	**60.2**	**86.3**	**62.0**	**9.4**	**2.1**	**1.3**
Qwen3.5-9B	Base	72.1	58.2	56.8	60.4	60.2	77.7	50.3	9.2	0.0	3.8
	w/ CPT	75.2	58.9	58.0	61.3	64.5	82.8	49.7	9.2	0.5	2.8
	w/ HyCPT	74.8	59.3	57.8	65.9	64.1	86.1	51.6	10.3	1.8	2.3
	N^2-MARS	**76.2**	**60.4**	**59.3**	**70.2**	**65.6**	**89.7**	**61.6**	**11.1**	**2.5**	**1.0**

3 EXPERIMENTS

We evaluate general capability (Section 3.1), long-context generation (Section 3.2), training hyperparameters (Section 3.4), and generated-token cost (Section 3.6). The backbones, benchmarks, baselines, metrics, and implementation details are provided in Appendix C.

3.1 PERFORMANCE ON GENERAL BENCHMARKS

Table 1 reports the results on nine general benchmarks, and we have the following observations. **Obs. 1. Native rollback improves multi-step reasoning.** N^2-MARS improves the base model in both math and planning benchmarks, suggesting that *the rollback command trained from open-domain text benefits tasks that require complex or multi-step reasoning capabilities*. **Obs. 2. General knowledge and capabilities are preserved in N^2-MARS.** The marginal change in MMLU, MMLU-PRO, and performance gain on IFEVAL indicate that N^2-MARS preserves the model's capability in general knowledge performance. **Obs. 3. Every stage contributes, and the RL stage contributes the most on reasoning.** Both our training stages, *i.e.*, HyCPT and GRPO, contribute to the model.

3.2 PERFORMANCE ON LONG-CONTEXT GENERATION BENCHMARKS

Table 2 reports the performance over baseline methods and N^2-MARS on three long-context generation benchmarks, from which we have the following observations. **Obs. 4. N^2-MARS is the best method on every backbone.** It has the best mean rank on all four backbones and is first or second in nearly every column. No single baseline comes close, *i.e.*, Best-of-N is the runner-up on three backbones and NL-Correct on Qwen3.5-4B. **Obs. 5. Training on the rollback corpus is insufficient to allow models to trigger the rollback command.** The CPT row shows what happens after the corpus alone: the shortest LONGPROC tier improves on the Qwen backbones, but the longest tier gets worse, and so do LONGGENBENCH and LONGBENCH-WRITE on every backbone. N^2-MARS, which learns to fire the command during generation, is above CPT in almost every column, and the lead grows as the output gets longer. The one exception is the completion rate on Llama-3.1-8B, which drops with CPT and does not come back, whereas Best-of-N keeps it.

3.3 ROLLBACK COMMAND ANALYSIS

Figure 4 reports the rollback command statistics on the three long-context benchmarks, GSM8K and IFEVAL, from which we have the following observations. **Obs. 6. Rollback frequency depends on the backbone family, and stronger backbones roll back less.** Qwen backbones fire the command more often than Llama backbones on every benchmark. Within a family, the larger model rolls back

Table 2: Performance comparison on long-context generation. Within each backbone, **bold** is the best and underline the second best per column, and **Rank** is the mean rank of the method over the nine columns.

	Method	LongProc			LongGenBench			LongBench-Write			Rank ↓
		0.5K↑	2K↑	8K↑	CR↑	STIC-1↑	STIC-2↑	S_l↑	S_q↑	$\bar{S}$↑	
Llama-3.2-3B	Base	1.7	0.1	0.0	**55.4**	12.8	6.7	26.9	24.7	25.8	3.9
	CPT	1.8	0.0	0.2	9.2	14.8	1.1	32.3	38.3	35.3	3.4
	NL-Correct	0.0	0.1	0.0	24.4	9.0	2.0	**40.2**	17.2	28.7	4.2
	Self-Refine	0.0	0.0	0.0	3.6	15.8	0.4	22.3	28.1	25.2	5.1
	Best-of-N	2.3	0.2	0.0	51.2	14.8	7.1	24.1	28.3	26.2	3.2
	N^2-MARS	**2.8**	**1.1**	**0.8**	53.6	**22.7**	**7.9**	40.1	**57.2**	**48.6**	**1.2**
Llama-3.1-8B	Base	3.7	2.0	0.0	58.3	12.4	6.6	19.5	33.6	26.5	3.5
	CPT	1.6	0.2	0.1	21.8	20.5	4.4	31.4	47.5	39.4	3.1
	NL-Correct	1.3	0.1	0.5	9.5	16.7	1.4	28.4	21.4	24.9	4.7
	Self-Refine	0.1	0.0	0.0	5.7	11.0	0.5	23.2	26.9	25.1	5.6
	Best-of-N	3.4	1.0	0.6	**73.7**	11.3	8.3	44.7	29.4	37.1	2.8
	N^2-MARS	**6.2**	**2.4**	**0.8**	69.8	**22.4**	**8.5**	**46.4**	**60.3**	**53.3**	**1.3**
Qwen3.5-4B	Base	5.0	1.0	3.1	26.1	39.5	10.0	71.3	**96.9**	84.1	3.3
	CPT	8.8	2.9	1.4	25.1	14.0	3.6	40.8	42.8	41.8	5.0
	NL-Correct	3.4	1.1	2.8	27.8	35.5	9.9	73.3	96.7	85.0	3.2
	Self-Refine	2.3	4.0	4.2	19.7	41.1	8.0	72.3	91.9	82.1	4.1
	Best-of-N	2.3	0.8	1.8	23.9	40.3	9.0	74.0	95.6	84.8	4.1
	N^2-MARS	**24.2**	**4.7**	**4.6**	**28.2**	**48.3**	**15.1**	**78.1**	94.4	**86.3**	**1.3**
Qwen3.5-9B	Base	2.4	0.5	4.0	31.2	46.5	14.5	74.9	96.9	85.9	3.3
	CPT	11.5	4.4	1.5	21.6	24.3	4.9	67.0	69.2	68.1	5.1
	NL-Correct	3.3	0.6	2.8	29.8	38.4	12.2	74.7	96.9	85.8	3.9
	Self-Refine	2.0	5.3	**4.9**	17.7	44.8	7.8	73.1	93.1	83.1	4.2
	Best-of-N	2.7	1.0	3.5	32.8	44.7	14.6	73.6	**97.5**	85.6	3.2
	N^2-MARS	**18.2**	**6.8**	4.7	**35.4**	**50.6**	**15.2**	**79.7**	97.2	**88.5**	**1.2**

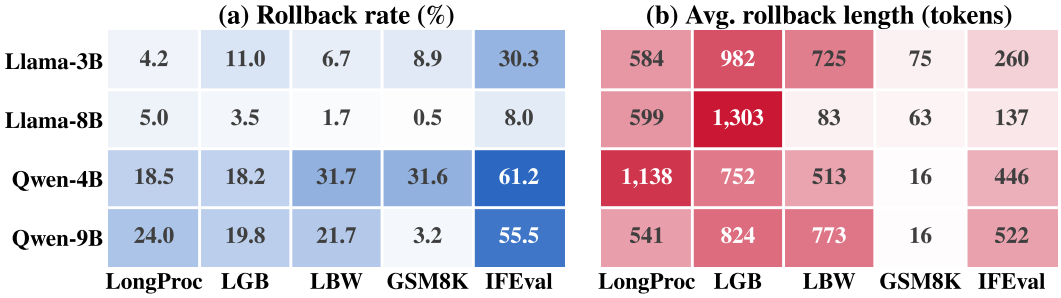

Figure 4: Rollback command statistics of N^2-MARS across benchmarks. (a) Percentage of responses that contains at least one rollback command, and (b) the average number of tokens rolled back per command. LGB: LongGenBench; LBW: LongBench-Write. Model labels abbreviate Llama-3.2-3B, Llama-3.1-8B, Qwen3.5-4B, and Qwen3.5-9B, respectively.

less, *e.g.*, Llama-3.1-8B is below Llama-3.2-3B on all five benchmarks, and Qwen3.5-9B is below Qwen3.5-4B on four. IFEval triggers the highest rate for every backbone, as constraint violations are easy to detect mid-generation and the command serves as a rewrite-to-comply mechanism. **Obs. 7. The rollback span scales with the output length of the task.** On the three long-context benchmarks, a single command deletes hundreds to over a thousand tokens, with LongGenBench the longest, whereas on GSM8K the Qwen backbones delete only 16 tokens, *i.e.*, roughly one arithmetic step, and IFEval sits in between. **Obs. 8. Frequency and span are decoupled.** Llama-3.1-8B rolls back on only 3.5% of LongGenBench responses but deletes the most tokens per command in the figure, while Qwen3.5-4B rolls back on 31.6% of GSM8K responses but only 16 tokens each, suggesting that *the model sets the scope of the command by the error rather than by a fixed budget*.

3.4 Hyperparameter Tuning

Figure 5 analyzes the three key hyperparameters that facilitate training the models, *i.e.*, (a) the learning rate of HyCPT, (b) the on-policy share of the data mixture, and (c) the ratio of rollback to clean tokens inside that on-policy share. We evaluate it on two metrics, *i.e.*, the rollback command

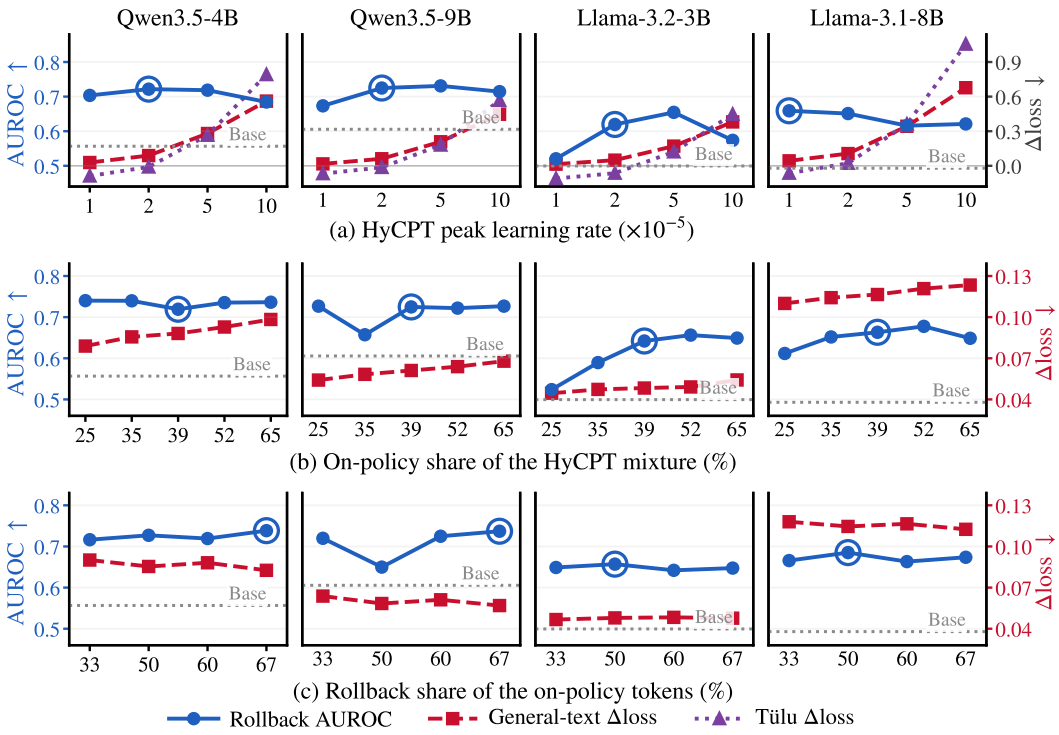

Figure 5: Three HyCPT sweeps, each trading rollback detection against retention.

Table 3: Judge models against human annotation.

Judge	Local	Cost ($/1M tokens)		RollbackTraj-Qwen3.5		RollbackTraj-Llama-3	
		Input	Output	Acc	F1	Acc	F1
`Qwen3.8-27B`	✓	n/a	n/a	92.6	93.1	93.0	94.4
`Claude Opus 5`	✗	5.00	25.00	94.6	95.2	97.0	98.6
`GPT-5.6 Sol`	✗	4.00	20.00	93.8	94.2	94.2	95.1

AUROC and loss on general text over the backbone, on a small on-policy sample set drawn from the rollback pool (500 samples) and the clean pool (500 samples), for a total of 1,000 samples. From these, we have the following observations. **(i) Detection keeps improving past the learning rate at which the backbone starts to pay.** Rollback AUROC rises steeply up to 2×10^{-5} and flattens or turns down beyond it, whereas the general-text loss keeps growing and the instruction loss turns from a gain into a penalty at the next step. We therefore take the largest rate that installs the primitive before instruction following turns against it, which is 2×10^{-5} on three backbones and 1×10^{-5} on Llama-3.1-8B. **(ii) A larger on-policy share is never free, and only the Llama family benefits from it.** The general-text loss grows monotonically with the share on every backbone, while the Qwen backbones are flat in AUROC across the range and the Llama backbones climb up to the second-largest share and fall back after it. The gain is small against the retention cost, so we keep the default share on all backbones, which holds Table 8 fixed across families. **(iii) The rollback-to-clean ratio is free of retention cost and is chosen per family.** The general-text loss barely moves across the row, so the ratio is selected on detection alone, and no backbone keeps the default, *i.e.*, both Qwen backbones peak at 2:1 and both Llama backbones at 1:1, which is the best of all their mixtures. Every other experiment uses the values selected here, which are summarized in Table 5.

3.5 Judge Model Selection

The judge J labels every sample of the rollback corpus and scores every open-domain prompt in reinforcement learning, so its agreement with human annotation bounds the quality of both stages. Table 3 compares the self-hosted `Qwen3.8-27B` with two proprietary judges, *i.e.*, `Claude Opus 5` and `GPT-5.6 Sol`, on 500 human-annotated samples from each released corpus in Table 6,

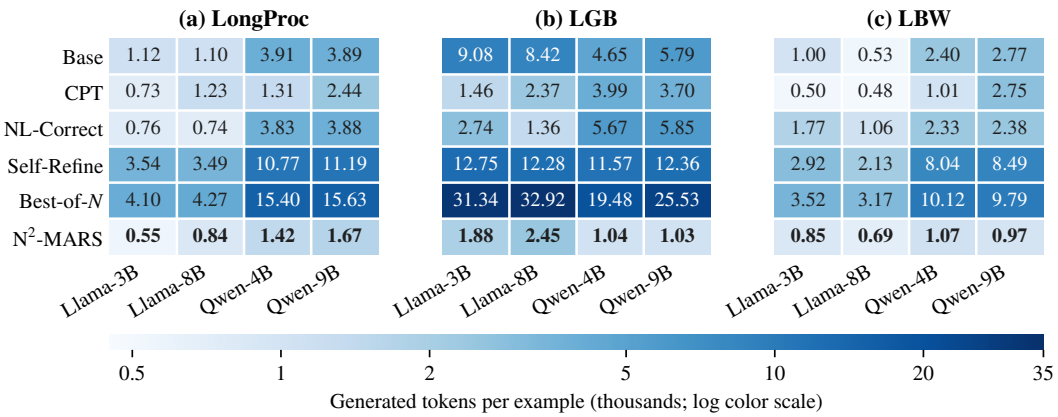

Figure 6: Generated-token cost across methods and backbones. Cells show mean tokens per example in thousands, including inferred selection tokens for Best-of-N ($N = 4$); colors share a logarithmic scale. Bold values denote N^2-MARS. Each cell averages 800 examples on LongProc (all available tiers pooled), 400 on LGB (LongGenBench), or 60 on LBW (LongBench-Write). Model abbreviations follow Figure 4.

(a) LongGenBench: workshop every 5 weeks from Week 13; mother's birthday on June 5

...#⋆# Week 23 (June 4th – June 10th): ~~Steady rain all week and a backlog of sequencing runs at the lab, so the weekend passed quietly indoors. #⋆# Week 24 (June 11th – June 17th):~~ <UNDO>45</UNDO> Mom turned sixty on Tuesday, June 5th, and we drove out for dinner between showers. Saturday was the parenting workshop again, the third one this year. ...#⋆# Week 24 (June 11th – June 17th): Paris fashion week ...

(b) LongProc: route from Lille to Bromsgrove; three of 49 lines point into Abingdon

From Lille, take a ferry to Abingdon. ~~From Abingdon, take a bus to Bromley. From Bromley, take a bus to Abingdon.~~ <UNDO>26</UNDO> From Abingdon, take a bus to Augsburg. From Augsburg, take a plane to Lecce. ...From Vladikavkaz, take a plane to Bromsgrove.

(c) LongBench-Write: first-person locked-room detective story, 2800 words

...the study door had been bolted from the inside, and the only key still hung from the dead man's belt. ...~~The butler admitted that he had let himself in with his spare key at midnight and found the study empty.~~ <UNDO>22</UNDO> The butler admitted that he had knocked at midnight and heard nothing, though the bolt was still drawn when the maid found the body at dawn.

Figure 7: Rollback in long-context generation, one illustrative trace per benchmark. Each trace shows a ~~deleted span~~, its command <UNDO>N</UNDO>, and the continuation on a real benchmark prompt, where N is the Qwen3.5 token count of the span shown.

where every judge receives the prompt in Figure 8 with the same prefix, completion, and reference. The self-hosted judge is within two points of accuracy and F1 of the proprietary judges on both corpora while carrying no API cost, so we adopt `Qwen3.8-27B` as the judge for all stages.

3.6 Generated-Token Cost

Figure 6 compares the mean number of generated tokens per example across baseline methods and N^2-MARS over three long-context generation benchmarks. Across all 12 settings, N^2-MARS generates 75.9–96.0% fewer tokens than Best-of-N and 67.8–91.7% fewer than Self-Refine. For example, on LongGenBench with Qwen3.5-9B, it generates 1,029 tokens per example, compared with 25,535 for Best-of-N and 12,362 for Self-Refine. This indicates N^2-MARS's token efficiency.

3.7 Case Study

Figure 7 illustrates three case studies from three long-context benchmarks, and we have the following observations: **Obs. 9. Errors surface only after the model has moved past them.** In (a), the missed

workshop and birthday become evident only when the model writes the header of the next week; in (b), the wrong turn is exposed two lines later when the route loops back to Abingdon; in (c), the contradiction with the bolted door appears only once the sentence is complete. This is the detection lag that the corpus construction in Section 2.1 trains for by placing the command at the end of the rejected completion rather than at the point of divergence. **Obs. 10. One command repairs a span of any size.** The three spans differ in length and content, *i.e.*, a diary entry plus a header, two route steps, and one sentence, yet each is removed by a single $\langle$UNDO$\rangle N\langle/$UNDO$\rangle$ and the repair continues in the same pass, whereas a single-token erase needs N commands and a post-hoc method needs a second pass over the whole draft.

4 Related Work

Iterative Refinement and Non-Monotonic Generation. A common direction to tackle error propagation is to critique and revise a finished draft [62, 19, 58, 40, 52, 13] or to select among complete generations [37, 6, 53], which regenerates at least one full response and thus scales in cost with the output length. Another line of work interleaves revision with decoding. For example, SequenceMatch [12] adds a backspace action with constant-time cache rollback, a reset token discards an unsafe response [78], a backtracking signal learned with RL on injected safety violations retracts a chosen number of tokens [50], REVERSE [65] and Self-Backtracking [71] revert hallucinated spans or reasoning steps, Stream of Revision [70] appends patches instead of deleting, and N-MARS [39] generalizes token-level erasure to reasoning tasks. More recently, a rollback is triggered at inference by an external monitor, *e.g.*, a guard model [25], perplexity or uncertainty signals [17, 66], or residual-stream activations [20]; reasoning models are trained with RL to erase a step, restart a chain, or recover from their own failed trajectories [60, 57, 49, 79, 75]; and editable canvases replace the append-only sequence altogether [14, 21]. Despite this progress, existing mechanisms roll back a fixed unit per command, depend on an external trigger, or learn the command on a single task from injected errors, whereas N^2-MARS learns a parameterized multi-token rollback from open-domain text, with learned control over when to fire and how far to roll back.

On-Policy Error Data and Long-Form Generation. Rollback is learnable only when the error, the rollback, and the correction co-occur in the training data [72, 16], and synthetically injected errors teach the injection pattern rather than self-correction, which motivates on-policy error data [30, 46, 64]. Recent recipes keep the model's own failed prefix and train the repair that follows it [36, 27, 10], retain negative samples through masked or reweighted losses instead of discarding them [7, 56, 38, 43], and train the verifier on the same on-policy outputs [76, 63], while theory shows that outcome-reward RL learns efficient backtracking where SFT on golden paths cannot [61]. Our corpus follows this principle, and HyCPT follows the replay recipe of continual pre-training [28] while masking erroneous tokens [41]. Moreover, long-context benchmarks have mostly measured the understanding of long inputs [3, 24, 74], whereas recent benchmarks measure the generation of long outputs [73, 45, 67, 4, 47, 69] and show that errors concentrate in the middle of a long output and that LLM judges become unstable on it [35, 8]. Methods for this regime scale the output length through RL or planning [68, 34, 54], whereas N^2-MARS targets the repair of a span once it has gone wrong, which is the value of rolling back.

5 Conclusion

In this work, we present N^2-MARS, a native non-monotonic autoregressive sequence modeling pipeline that we trained on four models from two model families, allowing models to roll back and repair their own generation through the rollback command $\langle$UNDO$\rangle N\langle/$UNDO$\rangle$ without task-specific supervision. Specifically, we propose a judged on-policy rollback corpus sampled from the model's own completions of open-domain documents, hybrid continued pre-training with source-specific loss masks, and reinforcement learning with outcome rewards that refines when the command fires. Comprehensive experiments on long-context generation and general benchmarks demonstrate N^2-MARS's performance across the evaluated tasks and backbones.

Reproducibility Statement

We release the source code of the sampling, judging, corpus building, training, and evaluation pipelines, the rollback corpus of every model family as text that is independent of any tokenizer (Table 6), and the twelve trained models of Table 4 together with their training provenance. The corpus construction is specified in Section 2.1 and Appendix B, the hyperparameters in Appendix A, and the benchmark protocols, prompts, generation budgets, and metric definitions in Appendix C. Every reported run records the revisions of the model, the dataset, and the evaluation harness, and all evaluations use fixed seeds.

Ethics Statement

This work studies a generation mechanism and uses only publicly available corpora and benchmarks under their respective licenses. The rollback corpus contains model-generated completions of public documents, including completions that a judge labels as erroneous, and it is released for research on self-correction rather than as a source of factual content. Rolled-back tokens are hidden from the visible output but remain available in the raw trajectory, which we expose in all released tooling so that the behavior of the model stays auditable.

Use of LLM Statement

In this work, we used generative AI tools to assist with implementation. LLM-generated code was verified and tested for correctness. We also used generative AI tools to polish the paper, and we have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

References

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

[2] Gregor Bachmann and Vaishnavh Nagarajan. The pitfalls of next-token prediction. In *ICML*, 2024.

[3] Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongBench: A bilingual, multitask benchmark for long context understanding. In *ACL*, 2024.

[4] Yushi Bai, Jiajie Zhang, Xin Lv, Linzhi Zheng, Siqi Zhu, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. LongWriter: Unleashing 10,000+ word generation from long context LLMs. In *ICLR*, 2025.

[5] Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. SmolLM-Corpus. https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus, 2024.

[6] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[7] Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Lifan Yuan, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. NFT: Bridging supervised learning and reinforcement learning in math reasoning. *arXiv preprint arXiv:2505.18116*, 2025.

[8] Junjie Chen, Yuxi Dong, Haitao Li, Weihang Su, Yujia Zhou, Min Zhang, Yiqun Liu, and Qingyao Ai. Benchmarking LLM-as-a-judge for long-form output evaluation. In *EMNLP*, 2026.

[9] 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, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[10] Qirui Chen, Renjie Pi, Jiahui Gao, and Lingpeng Kong. Reflective recovery: A self-supervised method for reasoning by learning from mistakes. *arXiv preprint arXiv:2609.19156*, 2026.

[11] 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. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[12] Chris Cundy and Stefano Ermon. SequenceMatch: Imitation learning for autoregressive sequence modelling with backtracking. In *ICLR*, 2024.

[13] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of ACL*, 2024.

[14] Sean Diab. Reviser: Revision-capable text generation via autoregressive cursor actions. *arXiv preprint arXiv:2609.20830*, 2026.

[15] Zeyu Gan, Yun Liao, and Yong Liu. Rethinking external slow-thinking: From snowball errors to probability of correct reasoning. In *ICML*, 2025.

[16] Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. Stream of search (SoS): Learning to search in language. In *COLM*, 2024.

[17] Jiaxun Gao, Him Wai Ng, and Z. Jane Wang. Once-More: Continuous self-correction for large language models via perplexity-guided intervention. In *ICLR*, 2026.

[18] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

[19] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *ICLR*, 2024.

[20] Manan Gupta and Dhruv Kumar. Latent phase-shift rollback: Inference-time error correction via residual stream monitoring and KV-cache steering. *arXiv preprint arXiv:2604.18567*, 2026.

[21] Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Variable length discrete flow matching with sequence-level edit operations. In *NeurIPS*, 2025.

[22] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *ICLR*, 2021.

[23] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS*, 2021.

[24] Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, Yang Zhang, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *COLM*, 2024.

[25] Xiaomeng Hu, Fei Huang, Chenhan Yuan, Junyang Lin, and Tsung-Yi Ho. CARE: Decoding-time safety alignment via rollback and introspection intervention. In *NeurIPS*, 2025.

[26] Luyang Huang, Shuyang Cao, Nikolaus Parulian, Heng Ji, and Lu Wang. Efficient attentions for long document summarization. In *NAACL*, 2021.

[27] Zhilin Huang, Hang Gao, Ziqiang Dong, Yuan Chen, Yifeng Luo, Chujun Qin, Jingyi Wang, Yang Yang, and Guanjun Jiang. Learning from your own mistakes: Constructing learnable micro-reflective trajectories for self-distillation. *arXiv preprint arXiv:2606.18844*, 2026.

[28] Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Anthony, Timothée Lesort, Eugene Belilovsky, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Transactions on Machine Learning Research*, 2024.

[29] Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. The Stack: 3 TB of permissively licensed source code. *Transactions on Machine Learning Research*, 2023.

[30] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. In *ICLR*, 2025.

[31] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *SOSP*, 2023.

[32] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tülu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[33] Hyunseok Lee, Seunghyuk Oh, Jaehyung Kim, Jinwoo Shin, and Jihoon Tack. ReVISE: Learning to refine at test-time via intrinsic self-verification. In *ICML*, 2025.

[34] Xuanyu Lei, Chenliang Li, Yuning Wu, Kaiming Liu, Weizhou Shen, Peng Li, Ming Yan, Fei Huang, Ya-Qin Zhang, and Yang Liu. Writing-RL: Advancing long-form writing via adaptive curriculum reinforcement learning. In *ACL*, 2026.

[35] Junjie Li, Xinrui Guo, Yuhao Wu, Roy Ka-Wei Lee, Hongzhi Li, and Yutao Xie. Lost in stories: Consistency bugs in long story generation by LLMs. In *Findings of ACL*, 2026.

[36] Lianrui Li, Dakuan Lu, Jiawei Shao, and Xuelong Li. ScRPO: From errors to insights. *arXiv preprint arXiv:2511.06065*, 2025.

[37] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *ICLR*, 2024.

[38] Zehua Liu, Shuqi Liu, Tao Zhong, and Mingxuan Yuan. RIFT: Repurposing negative samples via reward-informed fine-tuning. *arXiv preprint arXiv:2601.09253*, 2026.

[39] Tianyi Ma, Yiyue Qian, Yiyang Li, Zehong Wang, Yifan Ding, Zheyuan Zhang, Yan Liang, Chuxu Zhang, and Yanfang Ye. Non-monotonic autoregressive sequence model. In *ICML*, 2026.

[40] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *NeurIPS*, 2023.

[41] Amirkeivan Mohtashami, Martin Jaggi, and Sebastian Stich. Masked training of neural networks with partial gradients. In *AISTATS*, 2022.

[42] Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. OpenWebMath: An open dataset of high-quality mathematical web text. In *ICLR*, 2024.

[43] Rongcan Pei, Zhepei Wei, Shuyao Xu, Xinyu Zhu, Wei-Lin Chen, and Yu Meng. Negative self-distillation: Learning to reason by avoiding flaws. *arXiv preprint arXiv:2609.11699*, 2026.

[44] Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *NeurIPS Datasets and Benchmarks Track*, 2024.

[45] Chau Minh Pham, Simeng Sun, and Mohit Iyyer. Suri: Multi-constraint instruction following for long-form text generation. In *Findings of EMNLP*, 2024.

[46] Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. Recursive introspection: Teaching language model agents how to self-improve. In *NeurIPS*, 2024.

[47] Haoran Que, Feiyu Duan, Liqun He, Yutao Mou, Wangchunshu Zhou, Jiaheng Liu, Wenge Rong, Zekun Moore Wang, Jian Yang, Ge Zhang, Junran Peng, Zhaoxiang Zhang, Songyang Zhang, and Kai Chen. HelloBench: Evaluating long text generation capabilities of large language models. *arXiv preprint arXiv:2409.16191*, 2024.

[48] Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, Chloe Hillier, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling. In *ICLR*, 2020.

[49] Ankur Samanta, Akshayaa Magesh, Ayush Jain, Youliang Yu, Daniel Jiang, Kavosh Asadi, Kaveh Hassani, Paul Sajda, Jalaj Bhandari, and Yonathan Efroni. Credit assignment with resets in language model reasoning. *arXiv preprint arXiv:2605.25507*, 2026.

[50] Bilgehan Sel, Vaishakh Keshava, Phillip Wallis, Lukas Rutishauser, Ming Jin, and Dingcheng Li. Reinforcement learning with backtracking feedback. In *NeurIPS*, 2025.

[51] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[52] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

[53] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. In *ICLR*, 2025.

[54] Zechen Sun, Yuyang Sun, Zecheng Tang, Juntao Li, Wenpeng Hu, Wenliang Chen, Zhunchen Luo, Guotong Geng, and Min Zhang. IS-CoT: Breaking the long-form generation collapse via interleaved structural thinking. *arXiv preprint arXiv:2606.09709*, 2026.

[55] Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. LaMDA: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

[56] Xueyun Tian, Minghua Ma, Bingbing Xu, Nuoyan Lyu, Wei Li, Heng Dong, Zheng Chu, Yuanzhuo Wang, and Huawei Shen. Learning from mistakes: Negative reasoning samples enhance out-of-domain generalization. *arXiv preprint arXiv:2601.04992*, 2026.

[57] Pinzheng Wang, Shuli Xu, Juntao Li, Yu Luo, Dong Li, Jianye Hao, and Min Zhang. Re^2: Unlocking LLM reasoning via reinforcement learning with re-solving. In *ICLR*, 2026.

[58] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*, 2023.

[59] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. MMLU-Pro: A more robust and challenging multi-task language understanding benchmark. In *NeurIPS*, 2024.

[60] Ziliang Wang, Kang An, Xuhui Zheng, Faqiang Qian, Weikun Zhang, Cijun Ouyang, Jialu Cai, Yuhang Wang, and Yichao Wu. Erase to improve: Erasable reinforcement learning for search-augmented LLMs. In *ICLR*, 2026.

[61] Stanley Wei and Juno Kim. Provable benefits of RLVR over SFT for reasoning models: Learning to backtrack efficiently. *arXiv preprint arXiv:2606.22938*, 2026.

[62] Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. Generating sequences by learning to self-correct. In *ICLR*, 2023.

[63] Chen Henry Wu and Aditi Raghunathan. Self-trained verification for training- and test-time self-improvement. *arXiv preprint arXiv:2605.30290*, 2026.

[64] David X. Wu, Shreyas Kapur, Anant Sahai, and Stuart Russell. Synthetic error injection fails to elicit self-correction in language models. *arXiv preprint arXiv:2512.02389*, 2025.

[65] Tsung-Han Wu, Heekyung Lee, Jiaxin Ge, Joseph E Gonzalez, Trevor Darrell, and David M Chan. Generate, but verify: Reducing hallucination in vision-language models with retrospective resampling. In *NeurIPS*, 2025.

[66] Xianzong Wu, Xiaohong Li, Yuejun Guo, Xinyang Liu, Tianlin Li, Junjie Wang, and Qiang Hu. Do uncertainty signals help? A systematic study of uncertainty-aware decoding with rollback mechanisms. *arXiv preprint arXiv:2608.14653*, 2026.

[67] Yuhao Wu, Ming Shan Hee, Zhiqing Hu, and Roy Ka-Wei Lee. LongGenBench: Benchmarking long-form generation in long context LLMs. In *ICLR*, 2025.

[68] Yuhao Wu, Yushi Bai, Zhiqiang Hu, Roy Ka-Wei Lee, and Juanzi Li. LongWriter-Zero: Mastering ultra-long text generation via reinforcement learning. In *ICLR*, 2026.

[69] Zikai Xiao, Fei Huang, Jianhong Tu, Jianhui Wei, Wen Ma, Yuxuan Zhou, Jian Wu, Bowen Yu, Zuozhu Liu, and Junyang Lin. LongWeave: A long-form generation benchmark bridging real-world relevance and verifiability. In *Findings of EMNLP*, 2025.

[70] Chengran Yang, Zichao Wei, Heminghao Deng, Jinfeng Jiang, Zhensu Sun, Ting Zhang, Tianyi Wu, Ming Wen, and David Lo. Autoregressive, yet revisable: In decoding revision for secure code generation. *arXiv preprint arXiv:2602.01187*, 2026.

[71] Xiao-Wen Yang, Xuan-Yi Zhu, Wen-Da Wei, Ding-Chu Zhang, Jie-Jing Shao, Zhi Zhou, Lan-Zhe Guo, and Yu-Feng Li. Step back to leap forward: Self-backtracking for boosting reasoning of language models. *arXiv preprint arXiv:2502.04404*, 2025.

[72] Tian Ye, Zicheng Xu, Yuanzhi Li, and Zeyuan Allen-Zhu. Physics of language models: Part 2.2, how to learn from mistakes on grade-school math problems. In *ICLR*, 2025.

[73] Xi Ye, Fangcong Yin, Yinghui He, Joie Zhang, Howard Yen, Tianyu Gao, Greg Durrett, and Danqi Chen. LongProc: Benchmarking long-context language models on long procedural generation. In *COLM*, 2025.

[74] Howard Yen, Tianyu Gao, Minmin Hou, Ke Ding, Daniel Fleischer, Peter Izsak, Moshe Wasserblat, and Danqi Chen. HELMET: How to evaluate long-context language models effectively and thoroughly. In *ICLR*, 2025.

[75] Dongcheng Zhang, Yi Zhang, Yuxin Chen, An Zhang, Xiang Wang, and Chaochao Lu. Self-ReSET: Learning to self-recover from unsafe reasoning trajectories. *arXiv preprint arXiv:2605.08936*, 2026.

[76] Fuxiang Zhang, Jiacheng Xu, Chaojie Wang, Ce Cui, Yang Liu, and Bo An. Incentivizing LLMs to self-verify their answers. In *NeurIPS*, 2025.

[77] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How language model hallucinations can snowball. In *ICML*, 2024.

[78] Yiming Zhang, Jianfeng Chi, Hailey Nguyen, Kartikeya Upasani, Daniel M Bikel, Jason E Weston, and Eric Michael Smith. Backtracking improves generation safety. In *ICLR*, 2025.

[79] Xutong Zhao, Tengyu Xu, Xuewei Wang, Zhengxing Chen, Di Jin, Liang Tan, Yen-Ting Lin, Zishun Yu, Zhuokai Zhao, Yun He, Sinong Wang, Han Fang, Sarath Chandar, and Chen Zhu. Boosting LLM reasoning via spontaneous self-correction. *arXiv preprint arXiv:2506.06923*, 2025.

[80] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.

APPENDIX CONTENTS

A Implementation Details **17**

A.1 Model Card 17

A.2 Hybrid Continued Pre-Training Details 17

A.3 Reinforcement Learning Details 17

A.4 Inference 18

B Training Corpus **18**

B.1 Source Corpus 18

B.2 Trajectory Collection 18

B.3 LLM-as-Judge 19

B.4 Rollback Construction 19

B.5 Decontamination 20

C Experimental Setup **20**

A Implementation Details

A.1 Model Card

We release three models per backbone, as listed in Table 4, *i.e.*, N^2-MARS-<backbone>-HyCPT is trained with HyCPT only, and N^2-MARS-<backbone> is trained with HyCPT followed by GRPO, which is the final model. Moreover, we release the CPT version, named N^2-MARS-<backbone>-CPT, which is trained with the clean completions.

Table 4: Model card of N^2-MARS. Stages: the training stages applied, ✓ for applied. Tied: whether the input and output embeddings share parameters. Marker rows: vocabulary indices of ⟨UNDO⟩ and ⟨/UNDO⟩. Corpus: the family member from which the rollback corpus is sampled. The shaded rows are the models without a suffix.

Backbone	Model Name	Stages			Tied	Marker rows	Corpus sampled from
		CPT	HyCPT	GRPO			
Llama-3.2-3B	N^2-MARS-Llama-3.2-3B-CPT	✓			✓	128002 / 128003	Llama-3.2-3B
	N^2-MARS-Llama-3.2-3B-HyCPT		✓				
	N^2-MARS-Llama-3.2-3B		✓	✓			
Llama-3.1-8B	N^2-MARS-Llama-3.1-8B-CPT	✓			✗	128002 / 128003	Llama-3.2-3B
	N^2-MARS-Llama-3.1-8B-HyCPT		✓				
	N^2-MARS-Llama-3.1-8B		✓	✓			
Qwen3.5-4B	N^2-MARS-Qwen3.5-4B-CPT	✓			✓	248077 / 248078	Qwen3.5-4B
	N^2-MARS-Qwen3.5-4B-HyCPT		✓				
	N^2-MARS-Qwen3.5-4B		✓	✓			
Qwen3.5-9B	N^2-MARS-Qwen3.5-9B-CPT	✓			✗	248077 / 248078	Qwen3.5-4B
	N^2-MARS-Qwen3.5-9B-HyCPT		✓				
	N^2-MARS-Qwen3.5-9B		✓	✓			

A.2 Hybrid Continued Pre-Training Details

Marker Initialization. The marker embeddings are initialized as the average embedding of a set of anchor words,

$$\mathbf{E}[\langle\text{UNDO}\rangle] = \frac{1}{|\mathcal{W}_{\langle\text{UNDO}\rangle}|} \sum_{w \in \mathcal{W}_{\langle\text{UNDO}\rangle}} \frac{1}{|\text{tok}(w)|} \sum_{v \in \text{tok}(w)} \mathbf{E}[v], \tag{6}$$

where $\mathbf{E}$ is the embedding matrix, $\mathcal{W}_{\langle\text{UNDO}\rangle}$ is the anchor word set, and $\text{tok}(w)$ is the set of subword tokens of word w, and the anchor set of ⟨UNDO⟩ contains rollback words and the anchor set of ⟨/UNDO⟩ contains termination words, *e.g.*, *done*, *end*, *stop*, *close*, and *finish*.

HyCPT. Table 5 lists the hyperparameters, which are shared by all backbones except the peak learning rate of Llama-3.1-8B and the rollback-to-clean ratio, which differ by family as selected in Section 3.4. All parameters are trained. Sequences are packed into rows of 8,192 and 16,384 tokens, one source per row, where an example that would cross a row boundary is deferred to the next row, and the collator restarts the position indices at every example boundary. The builder verifies the realized ratio of clean to rollback tokens against the requested allocation with a 10% relative tolerance and aborts on a source that produces no examples.

A.3 Reinforcement Learning Details

The reward of Equation 5 is computed on the visible sequence $\Phi(\tau)$ with the constants of Table 5. A trajectory counts as *helped* when at least one visible answer immediately before a valid command is wrong and the final visible answer is correct, and the bonus is paid at most once per trajectory. A trajectory counts as *unnecessary* when it contains a valid command while every visible answer before a command is already correct, which is logged for the trigger analysis but does not enter the reward. The malformed-command term is charged once per trajectory when any interpreter event is rejected, *i.e.*, a missing or non-numeric argument, a missing closing marker, a zero span, or a span larger than the stack.

Design of the Outcome Matrix. The matrix ranks a correct output above an incorrect one for every value of c, pays the bonus β_{help} only for a verified wrong-to-right transition, pays $r_{0,1}$ only when the command rolled back a wrong visible answer rather than for any emitted command, and sets $r_{0,1} > r_{0,0}$, so that an attempted rollback is not penalized on prompts where no sampled trajectory succeeds.

Table 5: Hyperparameters of HyCPT and the reinforcement learning stage. The peak learning rate, the on-policy share, and the rollback-to-clean ratio are selected by the sweeps of Section 3.4, and a value that differs by family is listed per family.

HyCPT		Reinforcement learning	
Optimizer	AdamW	Algorithm	GRPO
Peak learning rate	2×10^{-5}	Learning rate	1×10^{-6}
Llama-3.1-8B	1×10^{-5}	Group size G	8
On-policy share	39%	KL coefficient β	0.04
Rollback-to-clean ratio	2:1 (Qwen) / 1:1 (Llama)	Sampling temperature	1.0
Warmup ratio	0.03	$r_{1,\cdot}$ (correct)	$+0.8$
Weight decay	0.1	β_{help} (wrong-to-right bonus)	$+0.2$
Gradient clipping	1.0	$r_{0,1}$ (wrong, valid command)	$+0.3$
Epochs	1	$r_{0,0}$ (wrong, no command)	-1.0
Per-device batch size	1	λ_{mal} (malformed command)	0.5
Gradient accumulation	8		
Packing lengths	8,192 / 16,384		
Precision / sharding	bf16 / ZeRO-2		

A.4 Inference

General benchmarks use the task-standard plain prompts of the pretrained checkpoints without a chat template, and long-context benchmarks use the chat template of each backbone. The interpreter is applied on token indices with special tokens preserved, and a rollback is never detected by matching decoded text.

B Training Corpus

The supervised corpus combines ordinary text, instruction-following conversations, and rollback trajectories with source-specific loss masks. We describe the corpus of the Qwen family below, and the corpora of the other families follow the same protocol with their own samplers. Table 6 summarizes the released corpora, one per model family, stored as text so that every member of the family can train on it.

Table 6: Dataset card of the rollback corpora. Sampler: the base model whose completions form the corpus. Judge: the model that labels each completion. Cap: the document length cap of the medium and the long tranche. Serves: the backbones trained on the corpus.

Corpus	Family	Sampler	Judge	Sources	Cap (medium / long)	Serves
RollbackTraj-Qwen3.5	Qwen3.5	Qwen3.5-4B	Qwen3.8-27B	7	4K / 16K	Qwen3.5-4B, Qwen3.5-9B
RollbackTraj-Llama-3	Llama 3	Llama-3.2-3B	Qwen3.8-27B	7	4K / 16K	Llama-3.2-3B, Llama-3.1-8B

B.1 Source Corpus

Our source corpus is drawn from seven sources, including FineWeb-Edu [44], Cosmopedia [5], English Wikipedia, the Python subset of The Stack [29], OpenWebMath [42], PG-19 [48], and GovReport [26]. The long-context source corpus comes from PG-19 and GovReport, using the reports rather than their summaries. These sources supply web prose, synthetic educational text, encyclopedic text, code, mathematics, books, and reports. General replay is streamed separately from FineWeb-Edu, and the instruction examples come from the Tülu 3 mixture [32].

B.2 Trajectory Collection

The document is cut at a fraction drawn uniformly from $[0.1, 0.8]$ of its length, fixed by a hash of the document identifier, so that the prefix $\mathbf{p}$ is the text before the cut and the reference completion $\mathbf{y}$ is the text after it.

B.3 LLM-AS-JUDGE

We adopt Qwen3.8-27B as the judge to annotate each completion that disagrees with the reference as WRONG or ACCEPTABLE, based on its consistency with the prefix and its internal plausibility. The verdict is read from the next-token logits of the two verdict words in a single forward pass, which makes the labels reproducible and the judge affordable at corpus scale. Figure 8 shows the judge instruction and the three per-example input fields. Table 7 reports the statistics per source for the Qwen3.5 and Llama 3 corpora.

Judge prompt

You are judging whether a language model's completion of a text is an acceptable alternative to a reference completion, or a mistake.

- **ACCEPTABLE** means the completion is factually consistent with the prefix and internally plausible, even if it differs in wording, detail, or emphasis from the reference, so similarity to the reference is **not** the standard.
- **WRONG** means the completion contradicts a fact established in the prefix, asserts something implausible, or is incoherent.

Weigh factual consistency with the prefix and internal plausibility of the completion itself, not how closely it resembles the reference.

Now judge this case:

Prefix: ⟨prefix⟩

Sampled completion: ⟨sampled completion⟩

Reference completion: ⟨reference completion⟩

Answer with exactly one word, WRONG or ACCEPTABLE.

Figure 8: Judge prompt. Judge instruction and per-example inputs (few-shot demonstrations omitted); highlighted fields are supplied per row as described in Appendix B.3.

Table 7: Statistics of the on-policy rollback corpus. One corpus per family, sampled with the model named in the first column. WRONG completions become rollback trajectories, and the clean pool contains the ACCEPTABLE completions together with those that never diverge from the reference. N is the length of the rejected completion that the command rolls back, in tokens of the sampler, over the WRONG rows of each source. Token counts are under the sampler's tokenizer, in millions, over all rows of the source.

		Rows					Tokens (M)			
	Source	Count	WRONG	ACCEPTABLE	Clean pool	Mean N	Prefix	Sampled	Reference	Total
Qwen3.5-4B	FineWeb-Edu	479,892	410,496	69,252	69,396	907	195.84	412.86	239.82	848.53
	Cosmopedia-v2	479,985	273,679	206,254	206,306	743	153.33	324.47	187.33	665.13
	Wikipedia	479,412	419,332	59,312	60,080	734	185.40	328.38	226.95	740.73
	The Stack (Python)	474,841	373,964	98,561	100,877	715	273.07	300.13	334.45	907.65
	OpenWebMath	479,823	400,854	77,863	78,969	820	292.44	377.44	357.58	1,027.46
	PG-19 (Medium)	28,602	27,354	1,247	1,248	1,021	52.36	29.20	63.95	145.51
	PG-19 (Long)	28,601	28,323	276	278	2,040	201.44	58.34	245.85	505.63
	GovReport	17,519	17,462	56	57	1,882	71.67	32.95	87.52	192.14
Llama-3.2-3B	FineWeb-Edu	479,697	340,527	138,875	139,170	699	190.71	280.36	233.53	704.60
	Cosmopedia-v2	479,981	240,631	239,310	239,350	666	150.57	277.57	183.96	612.10
	Wikipedia	479,060	347,450	130,294	131,610	591	176.76	232.80	216.36	625.92
	The Stack (Python)	477,791	349,871	123,725	127,920	697	258.91	278.36	317.00	854.27
	OpenWebMath	479,774	382,322	95,700	97,452	700	282.39	309.63	345.33	937.35
	PG-19 (Medium)	28,602	27,524	1,076	1,078	1,022	52.33	29.22	63.92	145.47
	PG-19 (Long)	28,602	28,275	326	327	2,038	201.02	58.29	245.35	504.67
	GovReport	17,518	17,413	102	105	1,715	70.09	30.01	85.58	185.68

B.4 ROLLBACK CONSTRUCTION

The sampled completion is cut at a randomly chosen sentence boundary for prose and line boundary for code before it is judged, so the text that the judge labels and the text that the command rolls back coincide. Rows without any boundary are dropped. The reference repair is truncated to at most 2,048 tokens. The span N in Equation 3 is recomputed with the tokenizer of the training target, and the

builder asserts that N equals the emitted length of $\hat{\mathbf{y}}$. The command therefore rolls back the sampled completion back to the document prefix before the repair begins.

Clean Completions. Medium-length clean examples retain the sampled completion without a command, and their truncation lengths are drawn from the rollback spans observed by the builder, so that the completion length is a weak cue for whether to roll back. For the long tranche, where the judge accepts few completions, the clean examples use the reference completion, truncated with lengths drawn from the long rollback pool. These negatives are reference text rather than model-generated text.

Mixture and Supervision. Table 8 summarizes the sources and their loss masks. The token allocation is 48% general replay, 13% instruction conversations, 11% medium rollback trajectories, 14% medium clean completions, 9% long rollback trajectories, and 5% long clean completions. General replay receives the full next-token objective. Instruction conversations are rendered with the chat template of the target, with supervision restricted to the final assistant response. For clean completions, the document prefix is masked and the completion and the end-of-sequence token are supervised. For rollback trajectories, every position before the command, including the rejected completion, is masked, and only the command, the repair, and the end-of-sequence token contribute to the objective.

Table 8: Sources of HyCPT and their loss masks. Shares are fractions of the token budget. Rollback and clean rows each combine a medium and a long tranche.

Source	Content	Share	Supervised positions ($M_t = 1$)
$\mathcal{D}_{\text{raw}}$	general web replay	48%	all tokens
$\mathcal{D}_{\text{inst}}$	instruction conversations	13%	final assistant turn
$\mathcal{D}_{\text{undo}}$	rollback trajectories (Equation 3)	11% + 9%	command $\langle\text{UNDO}\rangle N \langle/\text{UNDO}\rangle$, repair $\mathbf{y}$, $\langle\text{eos}\rangle$
$\mathcal{D}_{\text{clean}}$	paired clean completions	14% + 5%	completion, $\langle\text{eos}\rangle$

B.5 Decontamination

To ensure our training corpus contains no samples from the evaluation benchmarks, we measure the 13-gram overlap between the corpus and the test splits of all evaluation benchmarks, which is below 0.05% for every source. We simply drop these samples from our training corpus.

C Experimental Setup

Long-Context Generation Benchmarks. We evaluate long-context generation in natural mode, in which any correction is self-initiated.

- **LongProc** [73] requires the model to follow a procedure over a long input and to produce a long structured output that a deterministic checker scores. We use three task families, *i.e.*, `html_to_tsv`, `pseudo_to_code`, and `path_traversal`, at output tiers of 0.5K, 2K, and 8K tokens.
- **LongGenBench** [67] evaluates whether a model follows single, range, and periodic instructions throughout a generation of 16K tokens, and reports the completion rate and the instruction-following accuracy.
- **LongBench-Write** [4] evaluates whether the output length matches the requested length (S_l) and the quality of the writing (S_q).

General Benchmarks. We adopt nine general benchmarks covering knowledge and instruction following, code, mathematics, and planning.

- **MMLU** [22] and **MMLU-Pro** [59] measure knowledge and reasoning across academic subjects, evaluated with 5 shots, the latter with chain-of-thought, and aggregated by the number of test items.
- **IFEval** [80] measures instruction following with verifiable constraints, where we report the strict instruction-level accuracy.
- **HumanEval** [9] and **MBPP** [1] measure code generation with greedy pass@1, evaluated with 0 and 3 shots, respectively, inside an isolated sandbox.
- **GSM8K** [11] and **MATH** [23] measure mathematical reasoning with 8-shot and 4-shot chain-of-thought prompts and exact match, where MATH uses the full test set of 5,000 problems.

- **COUNT** contains 1,000 puzzles that require reaching a target with every given number used exactly once, and **SUDOKU** contains 1,000 9×9 puzzles from three difficulty tiers, where both are evaluated zero-shot and scored by exact validity.

Baseline Methods. We compare N^2-MARS with the following baseline methods:

- **Base** is the pretrained backbone, left untouched.
- **NL-Correct** appends an instruction to the base model that permits it to state that an earlier passage was wrong and to continue with a correction in natural language.
- **Self-Refine** [40] alternates between feedback and revision for a fixed number of rounds.
- **Best-of-N** [6] samples complete generations and selects one with the benchmark-agnostic self-evaluation prompt.

Evaluation. All benchmarks are evaluated with lm-evaluation-harness [18] and vLLM [31] using greedy decoding, and long-context benchmarks use a temperature of 1.0 and a top-p of 0.95.