TrajLong: Co-Designing Agentic and Long-Context Supervision for Mid-Training

Anonymous authors
Paper under double-blind review

Abstract

LLM agents for coding, search, and workplace tasks increasingly rely on long-context capabilities to effectively aggregate and reason over extended interaction histories. Recent work has incorporated agent trajectories into mid-training stage, drawing on their naturally long and interaction-rich structure. Yet how to organize the information within these trajectories into effective mid-training supervision remains underexplored. In this work, we investigate the relationship between long-context and agent atomic capabilities and introduce **TrajLong**, a novel framework that compiles trajectories into long-context training tasks with dense supervision, targeting three representative atomic capabilities: evidence grounding, cross-evidence aggregation, and temporal state maintenance. We mid-train Qwen3-14B-Base and Qwen3-30B-A3B-Base with data compiled by TrajLong, followed by supervised fine-tuning. Experiments on 6 long-context and 12 agent benchmarks demonstrate broad performance gains, with controlled ablations showing improvements over raw and masked trajectory baselines. Capability-level analyses further reveal task-dependent associations between long-context and agent atomic capabilities. These findings suggest that shared capability demands provide a principled basis for designing mid-training data to strengthen downstream agent capabilities.

1 Introduction

Large language models (LLMs) are increasingly expected to serve as agents in a wide range of real-world applications including coding, search, and workplace tasks (Yao et al., 2024; Liu et al., 2025; Shao et al., 2025). Completing these long-horizon user tasks requires LLMs to integrate information across many interaction turns, making effective use of long context essential to agent execution (Jimenez et al., 2024; Zhou et al., 2026; Su et al., 2026). Agent trajectories capture these extended interaction histories and provide a natural source of agent-oriented supervision. Recent work incorporates such supervision into mid-training, exposing base models to tool-use patterns and evidence-guided navigation before subsequent adaptation (Jiang et al., 2026; Su et al., 2025). Beyond teaching interaction patterns, these trajectories also offer rich long-context training material for developing the foundational capabilities required for agent execution.

The challenge is how to turn this rich but heterogeneous trajectory information into effective mid-training supervision. Applying next-token prediction to complete trajectories computes loss on both agent outputs and tool observations, introducing noisy tool responses into the training targets (Zeng et al., 2026; Chen et al., 2026; Yang et al., 2025b). To control these training signals, a common strategy is loss masking, which excludes selected tokens from the training loss. For example, masking observations and other non-assistant content concentrates supervision on agent reasoning and actions (Lu et al., 2026a; Zhou et al., 2026). However, masked content must still be processed as context, reducing training token efficiency. Figure 1 (left) contrasts full-trajectory and masked supervision. These choices determine where loss is applied, but do not by themselves organize long-range information dependencies into targeted learning tasks. ACC moves beyond loss allocation by compiling trajectory evidence into long-context question–answer examples for supervised fine-tuning, while retaining the original task question (Su et al., 2026). How to organize this information to systematically target distinct long-context capabilities remains open.

The information dependencies within these trajectories provide a concrete basis for designing such supervision. Similar search results require distinguishing a specific fact from related distractors;

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.

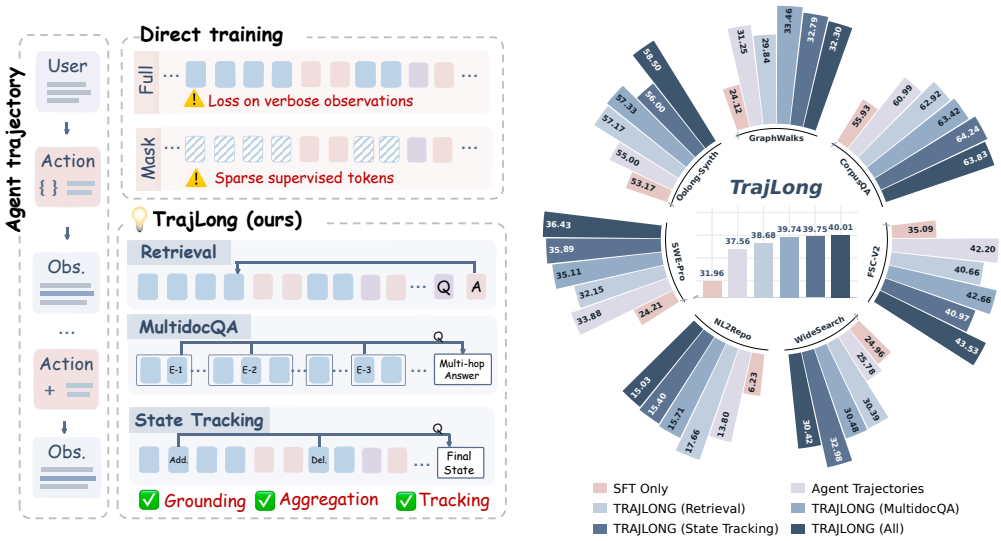


Figure 1: Overview of TrajLong. Left: Full-trajectory and masked mid-training strategies compared with TrajLong compilation. Right: Avg@3 scores of Qwen3-14B variants on seven long-context and agent benchmarks.

complementary observations must be combined across distant turns; and successive edits can invalidate earlier information about the same object. These dependencies resemble the retrieval, multi-hop reasoning, and aggregation demands of long-context evaluation (Hsieh et al., 2024), while adding the need to reconstruct evolving state (Wu et al., 2025). This perspective suggests that long-context reasoning and agentic execution may share capability primitives, providing a basis for organizing trajectory data around the information processing required by downstream tasks.

Figure 2 offers an exploratory view of this connection. Across four training settings, RULER-MultiKey ICL gains correlate positively with BrowseComp-ZH gains after the same SFT recipe ($\rho = 1.00$). It suggests a possible correlation between long-context capability and downstream agentic performance. This further motivates moving beyond aggregate long-context scores to investigate the capabilities relevant to agent execution. We therefore ask: ***How can the information dependencies within agent trajectories be organized into effective long-context supervision for agentic mid-training?***

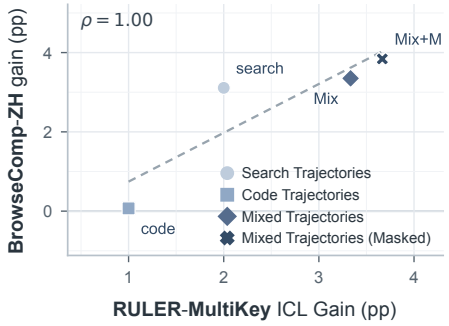


Figure 2: Correlation between RULER ICL gains and downstream BrowseComp-ZH gains across Qwen3-14B variants mid-trained with different agent trajectories.

We introduce **TrajLong**, a capability-targeted framework for agentic mid-training (Figure 1). It reorganizes agent trajectory context and constructs questions and target responses around the underlying information dependencies, targeting evidence grounding (Retrieval), cross-evidence aggregation (MultidocQA), and temporal state maintenance (State Tracking). We apply next-token prediction across all tokens in the compiled context, question, and target response.

We perform capability-targeted mid-training on Qwen3-14B-Base and Qwen3-30B-A3B-Base, followed by SFT, achieving gains over SFT-only baselines across diverse long-context benchmarks and search, tool-use, and coding agent benchmarks. Figure 1 (right) summarizes post-SFT performance of the Qwen3-14B variants on seven benchmarks. Matched-budget Qwen3-14B ablations further show that the combined variant outperforms both unmasked and masked raw-trajectory mixtures. Our analyses during mid-training and after SFT reveal overlapping long-context and agent gains with task-dependent learning dynamics. Evidence-based QA and dependency reasoning are associated with code understanding and planning, while state-oriented supervision may also support ag-

gregation and structured reasoning beyond temporal tasks, suggesting shared information-processing demands between long-context tasks and agent execution.

Our contributions are summarized as follows: **(1)** We introduce TrajLong, a trajectory compilation framework targeting evidence grounding, cross-evidence aggregation, and temporal state maintenance for long-context agentic mid-training. **(2)** We demonstrate broad downstream improvements on both 14B and 30B models across long-context reasoning, search, tool use, and coding agent tasks. **(3)** We characterize task-dependent correlations between long-context performance and agent atomic capabilities, and examine the utility of compiled supervision across mid-training and SFT.

2 Related Work

Long-context data construction. Natural long-form corpora are widely used for long-context training, but long documents are unevenly distributed across data sources (Fu et al., 2024). Existing work constructs long contexts through query-based grouping (Gao et al., 2025a), reference-linked document packing (Zhuang et al., 2025), and hard-negative insertion (Gao et al., 2025b). A complementary line develops synthetic QA supervision for retrieval and multi-evidence reasoning (Li et al., 2024a;b; Zhu et al., 2025; Wang et al., 2026a; Xu et al., 2026b). These approaches draw primarily on documents and webpages. ACC (Su et al., 2026) compiles evidence from search, software-engineering, and SQL agent trajectories into shuffled long contexts for question answering, demonstrating the value of agent trajectories as a training source. We further investigate capability-targeted trajectory compilation and the links between long-context capabilities and agent atomic capabilities.

Long-context mid-training. Mid-training, commonly implemented as continued pretraining (CPT), is widely used to develop long-context capabilities before downstream adaptation (Mo et al., 2025; Tu et al., 2025). Alongside positional adaptation for context-window extension (Ding et al., 2024; Shang et al., 2025; Lian et al., 2025), data-centric studies examine domain mixtures and sequence-length sampling (Fu et al., 2024; Runwal et al., 2026). Several training recipes combine mid-training with subsequent instruction tuning, using long-context data to strengthen context utilization (Zhao et al., 2024; Xu et al., 2025) or short-context data to maintain instruction-following capabilities (Xu et al., 2026a). Dependency-aware sample selection (Chen et al., 2024) and cross-repository code contexts (Paul et al., 2026) further highlight the importance of long-range structure in training data.

3 Methodology

TrajLong compiles agent trajectories into capability-targeted long-context training examples for mid-training. Figure 3 summarizes the trajectory preprocessing and compilation pipeline.

3.1 Problem Formulation

We study mid-training of a base model p_θ on trajectory-derived data before downstream supervised fine-tuning (SFT). A trajectory $\tau = (z_1, \ldots, z_N)$ records ordered messages, agent actions, and tool observations. Let $x = (x_1, \ldots, x_L)$ denote the token sequence of a training record and $m_t \in \{0, 1\}$ indicate whether token x_t is supervised. The next-token prediction objective is

$$\mathcal{L}(\theta; x, m) = -\frac{1}{\sum_{t=1}^{L} m_t} \sum_{t=1}^{L} m_t \log p_\theta(x_t \mid x_{<t}). \tag{1}$$

Unmasked mid-training on raw trajectories sets $m_t = 1$ at all valid prediction positions; masked mid-training excludes system prompts, user objectives, and tool observations from the loss while retaining them as context, supervising assistant reasoning, tool calls, and responses (Zhou et al., 2026). TrajLong retains the next-token prediction objective with full-token supervision and reorganizes trajectory information into capability-targeted training examples under a fixed token budget.

3.2 Trajectory Preprocessing

We collect agent trajectories from real-world software-development and workplace tasks and standardize them into a unified interaction format. We replace sensitive identifiers with consistent, type-

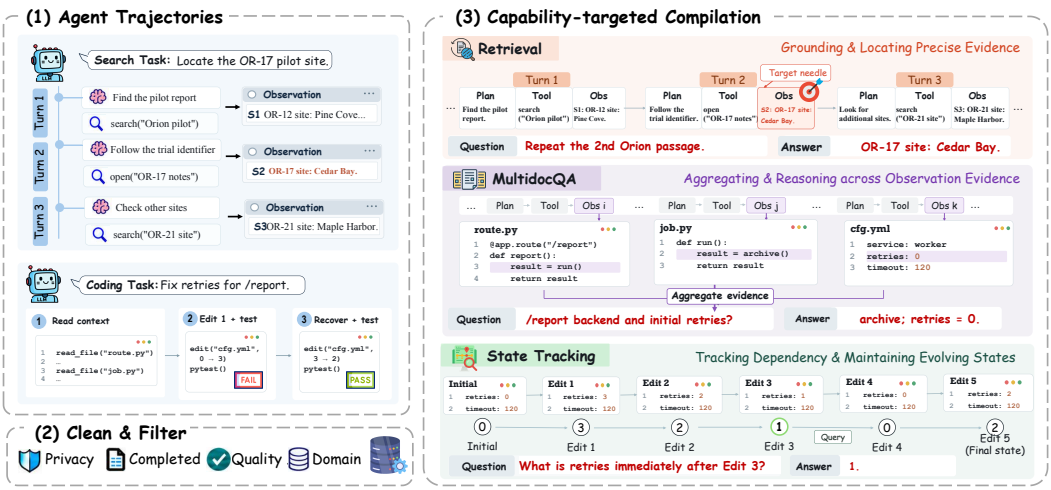


Figure 3: The TRAJLONG data construction pipeline. Search and coding agent trajectories are cleaned and filtered, then compiled into long-context training examples targeting evidence grounding (Retrieval), cross-evidence aggregation (MultidocQA), and temporal state maintenance (State Tracking).

appropriate surrogates to preserve within-trajectory references and exclude harmful tasks and trajectories with unresolved privacy or confidentiality risks.

We combine rule-based checks with LLM assessment to screen for format validity, task completeness, and interaction quality. We reject traces with excessive repetition, errors, or low-value actions, while treating error recovery and progressive exploration as high-value patterns. We also remove formatting noise introduced by agent harnesses, clarify user instructions, and refine tool-response content and formatting. Retained trajectories are assigned to code or search domains according to task intent and tool-use patterns. Detailed criteria are provided in Appendix A.1.

3.3 CAPABILITY-TARGETED COMPILATION

We target three representative long-context atomic capabilities: evidence grounding (***Retrieval***), cross-evidence aggregation (***MultidocQA***), and temporal state maintenance (***State Tracking***). Detailed construction procedures are provided in Appendix D.

Given a preprocessed trajectory τ, let $E(\tau) = \{o_i\}_i$ denote the evidence items extracted from its tool-call arguments and observations. A capability-specific compiler produces

$$\mathcal{C}_k(\tau) = \{(c_j, q_j, y_j)\}_j, \qquad k \in \{\text{Ret.}, \text{QA}, \text{ST}\}, \tag{2}$$

where c_j is the constructed context, q_j a newly constructed question, and y_j its target response derived from a teacher-generated solution. A reference answer a_j is used to construct or validate the final answer in y_j. We omit the example index j below.

Evidence grounding (Retrieval). This scheme targets precise retrieval of a specific content item from very long contexts. A single search call often returns multiple passages that share a topic but differ in their specific content. We retain the full preprocessed trajectory and treat these passages as natural *needles*, selecting one as the target while the others provide topically related distractors. Questions specify both topic and within-group position, making topical matching alone insufficient to identify the requested passage. For file interactions, questions specify path, read/write operation, and occurrence. Reference answers reproduce the selected evidence item $o_{i^\star}$ as rendered, $a = \text{render}(o_{i^\star})$. We check target uniqueness, answer–context alignment, and final-answer fidelity.

Cross-evidence aggregation (MultidocQA). This scheme targets multi-hop reasoning or aggregation of dispersed evidence (e.g., counting and comparison) across multiple sources. We organize file contents, retrieved documents, and execution results from different turns within a task, retaining source and version identifiers. An LLM constructs each question q and reference answer a from

a supporting set $S \subseteq E(\tau)$. We refine questions and their supporting evidence sets to emphasize joint use of evidence. A separate solver produces a solution for each question using the complete retained evidence collection. LLM-assisted checks review question answerability, ambiguity, and output requirements, followed by checks of final-answer agreement and solution consistency.

Temporal state maintenance (State Tracking). This scheme tracks the same object through repeated updates. We align file operations with tool responses and replay successful writes and edits e_t from a complete state anchor s_0, using $s_t = U(s_{t-1}, e_t)$. Anchors come from complete writes or reads checked for completeness. For *Snapshot* questions, the reference answer $a = s_{t_q}$ gives the requested state at t_q, covering pre-/post-edit, final, and suffix-rollback states. The context stops before the target file's first subsequent successful read, when present, to prevent copying the requested state from later observations. *Provenance* questions identify the operation type and file-specific index of a queried fragment's last write, first appearance, or removal. Partial updates must be composed while preserving untouched content. We check replay success and agreement with later complete reads when available, and validate teacher-generated final answers against the reconstructed references. We exclude QA groups where prior Snapshot states directly determine later answers.

3.4 Long-Context Mid-training

For mid-training, we combine a fixed 30B-token background corpus with 10B tokens of compiled trajectory data, yielding approximately 40B training tokens in total. The background corpus covers code and repositories, books and papers, web data, math and STEM, and instructions and logs. Compiled variants use Retrieval, MultidocQA, or State Tracking individually, or mix them at a 3:4:3 ratio by input-token count (TrajLong-All), with matched token budgets and optimization settings.

We group compiled examples sharing a context into training records, placing the context once before M QA pairs: $x = \text{serialize}(c, q_1, y_1, \ldots, q_M, y_M)$. We optimize Eq. (1) with $m_t = 1$ at all valid prediction positions, supervising the context and all questions and target responses. Mid-training uses 256K contexts and is followed by the same downstream SFT recipe across variants. Training details and corpus statistics are provided in Appendices D.1 and A.2.

4 Experimental Setup

Models and Mid-training. We conduct controlled experiments with Qwen3-14B-Base and additionally train TrajLong-*All* on Qwen3-30B-A3B-Base. Following Section 3.4, each mixture contains 30B background tokens and 10B trajectory-derived tokens. We train with ProLong (Gao et al., 2025c) at 256K sequence length, using a RoPE base of 1.6×10^7 without additional scaling. Mid-training uses AdamW, a peak learning rate of 4×10^{-5}, cosine decay, 10% warmup, and a global batch size of 64 packed sequences. Further training details are provided in Appendix D.1.

Downstream SFT. We construct our SFT data mixture from Nemotron-SFT-Agentic-v2, Nemotron-SFT-SWE-v3.5, and UltraChat-200K, with supervision restricted to assistant reasoning, responses, and tool calls. SFT uses 128K sequences while retaining the 256K positional configuration used for mid-training, with AdamW, a peak learning rate of 2×10^{-5}, cosine decay, 3% warmup, and a global batch size of 32 packed sequences. Main 14B comparisons use two effective epochs; the direct-SFT baseline starts from Qwen3-14B-Base with the same positional configuration and SFT recipe. Data mixture and training details appear in Appendices A.3 and D.1.

Evaluation. To probe capabilities learned during mid-training before SFT, we adapt long-context benchmarks into ICL tasks (**Long ICL**) targeting long-context atomic capabilities. Drawing on APT-Bench (Qin et al., 2025), we also implement **Agent ICL** to assess base models' agent atomic capabilities. These evaluations, together with general ICL, use greedy completion or candidate likelihood scoring, with a default generation limit of 1,024 tokens. After SFT, we evaluate long-context, search, tool-use, and coding agent benchmarks. Our downstream evaluation runs use task-level defaults of temperature 0.7, top-p 1.0, and a 65,536-token output cap per model call, subject to benchmark-specific settings. We report average and best-of-three metrics (Avg@3 and BoN@3), with score transformations and repeated-trial aggregation defined in Appendix C. More details about baselines and benchmarks are presented in Appendices B and E.3.

Table 1: Long-context benchmark performance (Avg@3 and BoN@3). Subscripts show changes relative to the corresponding Qwen3 model (pp). Bold marks the best score within each group.

Models	LongBench V2		MRCR		GraphWalks		GSM-Inf		CorpusQA	
	Avg@3	BoN@3	Avg@3	BoN@3	Avg@3	BoN@3	Avg@3	BoN@3	Avg@3	BoN@3
Frontier Models										
Gemini-2.5-Flash-Thinking	58.18	67.00	59.45	73.52	41.10	58.41	25.00	42.67	77.41	83.28
QwenLong L1.5-30B-A3B	45.04	54.69	55.12	61.51	39.24	44.16	39.56	64.33	80.14	88.45
GPT-OSS-120B	26.15	38.12	16.66	26.54	30.79	35.48	24.00	42.00	62.31	75.68
Our Models										
Qwen3-14B	26.21$_{+0.0}$	31.94$_{+0.0}$	19.16$_{+0.0}$	25.49$_{+0.0}$	27.02$_{+0.0}$	32.02$_{+0.0}$	3.89$_{+0.0}$	9.00$_{+0.0}$	56.43$_{+0.0}$	71.73$_{+0.0}$
- SFT *(w/o Mid-training)*	29.34$_{+3.1}$	44.71$_{+12.8}$	16.61$_{-2.6}$	26.24$_{+0.8}$	24.12$_{-2.9}$	30.18$_{-1.8}$	7.44$_{+3.6}$	17.67$_{+8.7}$	55.93$_{-0.5}$	73.86$_{+2.1}$
- TrajLong *(Retrieval)*	32.07$_{+5.9}$	45.31$_{+13.4}$	**21.60**$_{+2.4}$	**33.29**$_{+7.8}$	29.84$_{+2.8}$	37.02$_{+5.0}$	6.89$_{+3.0}$	16.67$_{+7.7}$	62.92$_{+6.5}$	78.72$_{+7.0}$
- TrajLong *(MultidocQA)*	**32.60**$_{+6.4}$	**47.50**$_{+15.6}$	19.93$_{+0.8}$	31.60$_{+6.1}$	**33.46**$_{+6.4}$	**39.61**$_{+7.6}$	7.22$_{+3.3}$	18.33$_{+9.3}$	63.42$_{+7.0}$	78.12$_{+6.4}$
- TrajLong *(State Tracking)*	31.87$_{+5.7}$	46.51$_{+14.6}$	19.36$_{+0.2}$	30.93$_{+5.4}$	32.79$_{+5.8}$	37.65$_{+5.6}$	**12.22**$_{+8.3}$	**26.00**$_{+17.0}$	**64.24**$_{+7.8}$	78.42$_{+6.7}$
TrajLong-14B-All	31.87$_{+5.7}$	46.91$_{+15.0}$	20.18$_{+1.0}$	31.27$_{+5.8}$	32.30$_{+5.3}$	38.20$_{+6.2}$	6.67$_{+2.8}$	16.33$_{+7.3}$	63.83$_{+7.4}$	**79.94**$_{+8.2}$
Qwen3-30B-A3B	23.15$_{+0.0}$	29.34$_{+0.0}$	18.06$_{+0.0}$	23.86$_{+0.0}$	27.29$_{+0.0}$	30.71$_{+0.0}$	4.67$_{+0.0}$	10.33$_{+0.0}$	56.53$_{+0.0}$	72.64$_{+0.0}$
- SFT *(w/o Mid-training)*	27.94$_{+4.8}$	42.32$_{+13.0}$	13.55$_{-4.5}$	23.16$_{-0.7}$	25.19$_{-2.1}$	32.52$_{+1.8}$	5.00$_{+0.3}$	12.00$_{+1.7}$	53.60$_{-2.9}$	73.86$_{+1.2}$
TrajLong-30B-A3B-All	**32.01**$_{+8.9}$	**45.33**$_{+16.0}$	**19.36**$_{+1.3}$	**28.66**$_{+4.8}$	**29.90**$_{+2.6}$	**36.56**$_{+5.9}$	**12.44**$_{+7.8}$	**29.33**$_{+19.0}$	**60.79**$_{+4.3}$	**78.72**$_{+6.1}$

5 Experimental Results

5.1 Main Results

In this section, we train different LLMs with TrajLong data in the mid-training stage, comparing with different frontier long-context LLMs. More results can be found in Appendix E.

Capability-Targeted Compilation Improves Long-Context Performance. Table 1 shows distinct long-context strengths across the three compilation schemes. Among the Qwen3-14B variants, *Retrieval* leads on MRCR, improving Avg@3 by 4.99 percentage points over SFT without mid-training. *MultidocQA* performs best on LongBench V2 and GraphWalks, while *State Tracking* achieves the strongest GSM-Infinite results, with a 4.78-point Avg@3 gain over the same baseline. These task-dependent differences are consistent with the distinct information dependencies emphasized during compilation. Combining the three capabilities in TrajLong-*All* improves both Avg@3 and BoN@3 over SFT-only on nearly all benchmarks for Qwen3-14B and Qwen3-30B-A3B. These findings suggest that compiling agent trajectories into tasks targeting atomic capabilities can improve general long-context performance, with each scheme exhibiting distinct task-specific strengths.

Long-Context Gains Transfer to Downstream Agentic Tasks. Table 2 shows improvements across search, tool use, and coding under the same downstream SFT recipe. For Qwen3-14B, each specialized compilation scheme outperforms SFT without mid-training on all 12 agent benchmarks. Combining the three schemes in TrajLong-14B-All yields the strongest SWE-Verified and SWE-Pro results among the 14B and 30B variants. These results suggest that capability-targeted long-context mid-training supports downstream transfer across search, tool use, and coding. The benefits extend beyond a fixed correspondence between compilation schemes and agent categories, suggesting that shared long-context capabilities provide a transferable foundation for diverse agentic tasks.

5.2 Ablation Study

Table 3 compares data construction under matched mid-training token budgets, a fixed background corpus, and the same downstream SFT recipe. Search trajectories outperform code trajectories on Oolong-Synth and SkillsBench, whereas code trajectories are stronger on GraphWalks and the two SWE benchmarks. Capability-targeted compilation yields broader improvements. TrajLong-*All* outperforms both unmasked raw trajectories (*Combined Trajectories*) and masked raw trajectories (*Combined Trajectories (with Mask)*) on all seven benchmarks. Relative to the unmasked baseline, it improves Avg@3 by 2.03 points on CorpusQA and 2.66 points on FSC-V2, with additional gains on tool-use and coding tasks. These results suggest that organizing trajectory information into capability-targeted long-context tasks provides benefits beyond source mixing or loss masking alone.

Table 2: Performance on search, tool-use, and coding agent benchmarks (Avg@3). Bold marks the best score within each group. Benchmark details and BoN@3 results are in Appendix B.3 and E.2.

Models	Search					Tool Use			Coding			
	BC	BC-ZH	WideSearch	FSC-V2	SealQA	MCP-Atlas	SkillsBench	WebArena	TB 2.1	NL2Repo	SWE-Verified	SWE-Pro
Frontier Models												
Gemini-2.5-Flash-Thinking	12.72	27.89	30.84	34.48	24.02	19.87	7.41	39.43	17.75	12.40	48.60	24.08
QwenLong L1.5-30B-A3B	2.98	11.07	35.27	20.36	16.58	32.73	1.39	27.17	1.87	7.71	31.60	10.94
GPT-OSS-120B	25.62	24.22	36.46	41.09	22.52	39.40	-	31.40	12.36	6.88	53.00	27.13
Our Models												
Qwen3-14B	1.87	10.45	29.91	22.81	13.87	11.93	0.28	19.80	0.37	5.59	18.07	4.29
- SFT *(w/o Mid-training)*	12.53	18.27	24.96	35.09	24.32	42.40	6.48	27.32	7.12	6.23	42.13	24.21
- TrajLong *(Retrieval)*	13.14	18.75	30.39	40.66	26.13	45.80	9.26	**30.32**	**15.73**	**17.66**	50.87	32.15
- TrajLong *(MultidocQA)*	**13.93**	18.34	30.48	42.66	24.68	**46.73**	6.94	29.88	9.36	15.71	52.13	35.11
- TrajLong *(State Tracking)*	12.72	**19.58**	**32.98**	40.97	**31.35**	45.00	**12.04**	28.42	14.61	15.40	53.07	35.89
TrajLong-14B-All	11.77	19.38	30.42	**43.53**	24.32	44.00	11.11	29.48	14.23	15.03	**54.40**	**36.43**
Qwen3-30B-A3B	2.00	10.59	26.62	20.46	14.77	20.53	0.93	12.32	1.50	7.08	25.47	9.48
- SFT *(w/o Mid-training)*	10.27	16.54	22.51	38.31	**25.41**	37.67	4.63	25.49	1.87	9.57	36.53	23.07
TrajLong-30B-A3B-All	**11.35**	**20.07**	**28.16**	**41.69**	25.23	**40.53**	**7.41**	**26.13**	**12.36**	**13.79**	**51.07**	**34.34**

Table 3: Ablation Study (Avg@3) of the mid-training data construction on long-context and agent benchmarks. All variants use the same Qwen3-14B backbone, downstream SFT recipe, and evaluation protocol. The trajectory baselines use raw search trajectories, raw code trajectories, or their combination, with an additional masked variant. All mid-training runs use the same token budget.

Mid-training Variant	Long-Context			Agent			
	Oolong-Synth	GraphWalks	CorpusQA	FSC-V2	SkillsBench	SWE-Verified	SWE-Pro
TrajLong-*All*	58.50$_{+0.0}$	32.30$_{+0.0}$	63.83$_{+0.0}$	43.53$_{+0.0}$	11.11$_{+0.0}$	54.40$_{+0.0}$	36.43$_{+0.0}$
SFT only	53.17$_{-5.33}$	24.12$_{-8.18}$	55.93$_{-7.90}$	35.09$_{-8.44}$	6.48$_{-4.63}$	42.13$_{-12.27}$	24.21$_{-12.22}$
Search Trajectories	57.33$_{-1.17}$	31.58$_{-0.72}$	61.80$_{-2.03}$	40.05$_{-3.48}$	9.72$_{-1.39}$	50.27$_{-4.13}$	31.37$_{-5.06}$
Code Trajectories	55.33$_{-3.17}$	32.77$_{+0.47}$	61.80$_{-2.03}$	39.49$_{-4.04}$	8.80$_{-2.31}$	52.40$_{-2.00}$	34.38$_{-2.05}$
Combined Trajectories	56.67$_{-1.83}$	31.38$_{-0.92}$	61.80$_{-2.03}$	40.87$_{-2.66}$	9.26$_{-1.85}$	53.60$_{-0.80}$	35.93$_{-0.50}$
Combined Trajectories (with Mask)	55.00$_{-3.50}$	31.25$_{-1.05}$	60.99$_{-2.84}$	42.20$_{-1.33}$	6.48$_{-4.63}$	53.20$_{-1.20}$	33.88$_{-2.55}$

5.3 Analysis

5.3.1 Capability-Aligned ICL Learning Dynamics

Figure 4 shows that long-context ICL gains generally rise before leveling off. *Retrieval* reaches its highest observed MRCR gain at the first evaluated checkpoint (approximately 5B tokens), whereas *MultidocQA* on BrowseCompLong and *State Tracking* on GraphWalks + LongBench V2 peak later. On Agent ICL, *MultidocQA* leads multi-hop replanning throughout training, while *State Tracking* consistently outperforms Raw Trajectories on the tool-selection/argument-filling composite.

These results suggest that **targeted trajectory supervision yields overlapping gains across long-context and agent tasks, but with task-dependent learning dynamics**. *Retrieval* improves MRCR sharply early in training, whereas *MultidocQA* on BrowseCompLong and *State Tracking* on GraphWalks + LongBench V2 achieve their largest observed gains later, despite intermediate fluctuations. The timing of these gains also differs between long-context tasks and agent ICL.

5.3.2 Long–Agent Associations across Atomic Capabilities

We examine how mid-training ICL gains on proxies for the three long-context atomic capabilities correlate with gains in individual agent atomic capabilities (Figure 5). This Spearman analysis uses five training variants (Retrieval, MultidocQA, State Tracking, All, and Raw Trajectories), with gains relative to unmasked Raw Trajectories. Final-checkpoint ICL gain profiles for the individual training variants are reported in Appendix E.4.

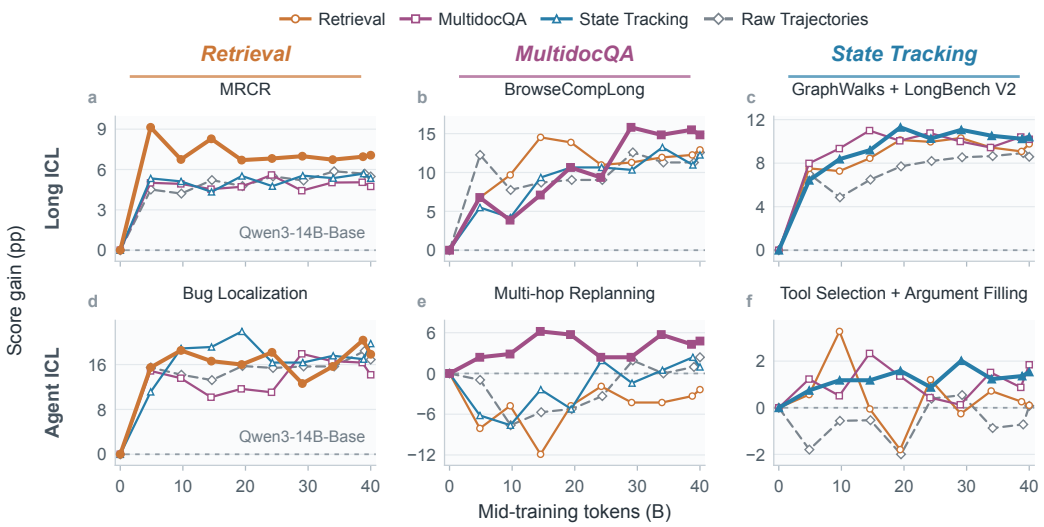


Figure 4: ICL score gains over Qwen3-14B-Base on long-context (top) and agent (bottom) tasks during mid-training (pp). Training steps are scaled to 40B tokens per run.

Within the *Grounding* group, RULER-Multivalue is most positively associated with Tool Selection, consistent with selecting task-relevant information from competing candidates.

The *Evidence-based QA* group, comprising BrowseCompLong and RULER-QA1, shows more consistent positive associations with Code Summary, Bug Fixing, Planning Preference, and Multi-hop Replanning across both benchmarks. These patterns are consistent with shared demands for using and aggregating contextual evidence when interpreting code, evaluating repairs, or choosing actions based on interaction history.

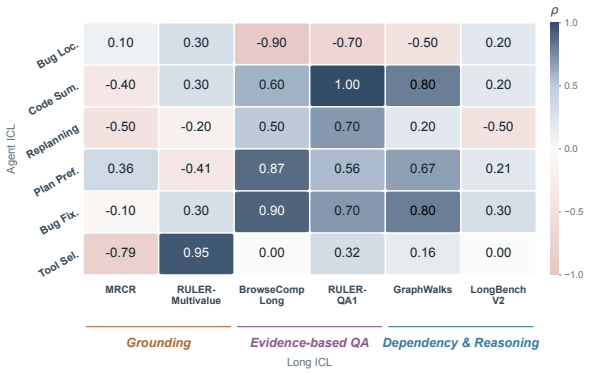


Figure 5: Spearman correlations between long-context and agent ICL gains across five Qwen3-14B variants at final mid-training checkpoints, relative to unmasked Raw Trajectories.

Within the *Dependency & Reasoning* group, GraphWalks correlates strongly with Code Summary and Bug Fixing (both $\rho = 0.80$), followed by Planning Preference ($\rho = 0.67$). This pattern suggests shared demands for tracing dependencies and understanding program behavior. Together, **evidence-based QA and dependency reasoning exhibit overlapping associations with code understanding and planning**, while the weaker or divergent patterns on MRCR and LongBench V2 suggest that these relationships depend on the specific information-processing demands of each benchmark.

5.3.3 Capability-Specific Gains after Downstream SFT

Figure 6 compares the compiled variants with unmasked Raw Trajectories under the same downstream SFT recipe. *Retrieval* yields the largest gains on MRCR (+1.79 pp) and WebArena, alongside improvements on MCP-Atlas, but regresses on GraphWalks and both SWE benchmarks. **Its retrieval advantage therefore coexists with benefits on interactive tasks**, without extending uniformly to dependency reasoning or software engineering.

MultidocQA leads on GraphWalks and LongBench V2, while also improving CorpusQA. Its strongest agent gains occur on BrowseComp and MCP-Atlas (+2.93 pp). These paired advantages are consistent with shared demands for **combining contextual evidence and tracing dependencies in long-context reasoning, information seeking, and multi-step tool composition**.

State Tracking achieves the largest gain on GSM-Inf (+4.89 pp) and CorpusQA. On agent tasks, it leads on WideSearch, SealQA, and SkillsBench. The CorpusQA–WideSearch pairing suggests a shared need to consolidate many facts into a coherent answer, while the GSM-Inf gain is consistent with tracking intermediate quantities through dependent computations. Thus, **state-oriented supervision may support aggregation and structured reasoning beyond explicitly temporal tasks**.

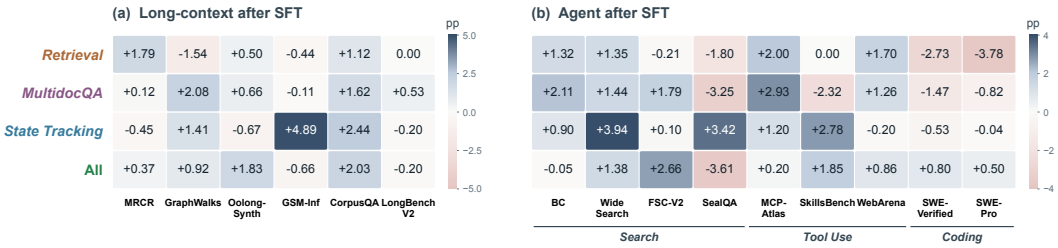


Figure 6: Post-SFT Avg@3 gains of Qwen3-14B variants on long-context and agent benchmarks, relative to unmasked Raw Trajectories under the same SFT recipe. Panels use separate color scales.

5.3.4 TrajLong across Training Stages

We further analyze whether TrajLong-compiled data also benefit from direct SFT without mid-training. We convert them to dialogue format and mix them with the original SFT data at 3:7, preserving the total SFT token amount. Table 4 shows gains over *None* on all six benchmarks, indicating that **capability-targeted compilation of TrajLong remains useful under assistant-only SFT**. Under the same two-epoch direct-SFT recipe, 3:7 mixture ratio, and token budget, **TrajLong outperforms ACC (Su et al., 2026) on most long-context benchmarks and all three evaluated agent benchmarks**, including a 3.89 pp advantage on SWE-Pro.

Across one to three SFT epochs after mid-training, long-context gains over *None* remain broadly stable, while agent performance shows modest overall improvement with additional epochs. These results suggest that extending SFT preserves the long-context benefits of our mid-training recipe while providing incremental improvements in downstream agent performance.

Table 4: Training-stage and SFT-epoch comparisons on Qwen3-14B (Avg@3). *None* denotes SFT on the original data only. Subscripts show changes from *None* (pp); bold marks column maxima.

Setting	SFT Epoch	Long-Context			Agent		
		Oolong-Synth	GraphWalks	CorpusQA	BC	NL2Repo	SWE-Pro
None	2	53.17$_{+0.0}$	24.12$_{+0.0}$	55.93$_{+0.0}$	12.53$_{+0.0}$	6.23$_{+0.0}$	24.21$_{+0.0}$
ACC (SFT)	2	56.00$_{+2.83}$	30.62$_{+6.50}$	57.89$_{+1.96}$	10.16$_{-2.37}$	7.00$_{+0.77}$	27.22$_{+3.01}$
TrajLong (SFT)	2	56.83$_{+3.66}$	28.04$_{+3.92}$	59.27$_{+3.34}$	13.56$_{+1.03}$	9.40$_{+3.17}$	31.11$_{+6.90}$
TrajLong (Mid-train+SFT)	1	55.83$_{+2.66}$	31.64$_{+7.52}$	**64.44**$_{+8.51}$	11.48$_{-1.05}$	14.16$_{+7.93}$	34.15$_{+9.94}$
	2	**58.50**$_{+5.33}$	**32.30**$_{+8.18}$	63.83$_{+7.90}$	11.77$_{-0.76}$	**15.03**$_{+8.80}$	36.43$_{+12.22}$
	3	**58.50**$_{+5.33}$	32.02$_{+7.90}$	63.22$_{+7.29}$	**14.67**$_{+2.14}$	14.81$_{+8.58}$	**37.21**$_{+13.00}$

6 Conclusion

We presented TrajLong, a framework that compiles agent trajectories into long-context supervision targeting evidence grounding, cross-evidence aggregation, and temporal state maintenance. Experiments on Qwen3-14B and 30B show gains across long-context and agent benchmarks, compared with both raw and masked trajectories. Our analyses reveal task-dependent learning dynamics and correlations between long-context performance and agent atomic capabilities, with evidence-based QA and dependency reasoning associated with code understanding and planning. State-oriented supervision may also support aggregation and structured reasoning beyond temporal tasks. Together, these findings support organizing trajectory information around long-context capabilities as a basis for effective agentic mid-training rather than being left entirely to post-training.

AI Use Statement

We used generative AI tools to assist with manuscript drafting, revision, and language editing, as well as the interpretation and presentation of experimental results. As part of the research pipeline, we used LLMs for trajectory screening, synthetic question construction and refinement, solution generation, and benchmark-specific answer evaluation. These uses and the corresponding quality checks are described in Appendices A, C, and D. We reviewed the AI-assisted content and checked the associated code and experimental results. We take full responsibility for the final content of this work, including text, claims, and artifacts produced with the assistance of generative AI.

Ethics Statement

This work studies capability-targeted mid-training for long-context understanding and agentic tasks. Our source trajectories were collected with users' knowledge and authorization, with the necessary permissions and licenses for research use. As detailed in Appendix A.1, we replace detected sensitive identifiers with consistent synthetic surrogates and apply organization-level exclusion, semantic privacy screening, and harmful-task filtering. We acknowledge that stronger agent capabilities may be misused and that training on existing trajectories may preserve or amplify societal biases. Downstream applications should therefore include task-specific safety evaluation, access controls, and human oversight. Research-use authorization does not imply permission to publicly redistribute the source trajectories.

Reproducibility Statement

We describe trajectory preprocessing and capability-targeted compilation in Section 3, with construction and validation procedures in Appendix D. Appendix A documents data sources, mixture composition, and benchmark decontamination. Training configurations and token accounting are reported in Section 4 and Appendix D.1. Benchmark definitions, ICL adaptations, and evaluation protocols are provided in Appendices B and C. Representative compiled examples appear in Appendix F.

References

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

Yushi Bai, Shangqing Tu, Jiajie Zhang, Hao Peng, Xiaozhi Wang, Xin Lv, Shulin Cao, Jiazheng Xu, Lei Hou, Yuxiao Dong, et al. Longbench v2: Towards deeper understanding and reasoning on realistic long-context multitasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3639–3664, 2025.

Chaithanya Bandi, Razvan-Gabriel Dumitru, Ben Hertzberg, Divyansh Agarwal, Geobio Boo, Tejas Polakam, Sami Hassaan, Jeff Da, HiJae Kim, Vipul Gupta, et al. Mcp-atlas: A large-scale benchmark for tool-use competency with real mcp servers. *arXiv preprint arXiv:2602.00933*, 2026.

Amanda Bertsch, Adithya Pratapa, Teruko Mitamura, Graham Neubig, and Matthew R Gormley. Oolong: Evaluating long context reasoning and aggregation capabilities. *arXiv preprint arXiv:2511.02817*, 2025.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: reasoning about physical commonsense in natural language. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pp. 7432–7439. AAAI Press, 2020. doi: 10.1609/AAAI.V34I05.6239. URL https://doi.org/10.1609/aaai.v34i05.6239.

Longze Chen, Ziqiang Liu, Wanwei He, Yinhe Zheng, Hao Sun, Yunshui Li, Run Luo, and Min Yang. Long context is not long at all: A prospector of long-dependency data for large language models. In

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pp. 8222–8234, 2024.

Mouxiang Chen, Lei Zhang, Yunlong Feng, Xuwu Wang, Wenting Zhao, Ruisheng Cao, Jiaxi Yang, Jiawei Chen, Mingze Li, Zeyao Ma, Hao Ge, Zongmeng Zhang, Zeyu Cui, Dayiheng Liu, Jingren Zhou, Jianling Sun, Junyang Lin, and Binyuan Hui. Swe-universe: Scale real-world verifiable environments to millions. *CoRR*, abs/2602.02361, 2026.

Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, Aoyan Li, Bo Li, et al. Fullstack bench: Evaluating llms as full stack coders. *arXiv preprint arXiv:2412.00535*, 2024.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

DeepSeek-AI. Deepseek-v4: Towards highly efficient million-token context intelligence. *CoRR*, abs/2606.19348, 2026.

Xiang Deng, Jeff Da, Edwin Pan, Yannis Yiming He, Charles Ide, Kanak Garg, Niklas Lauffer, Andrew Park, Nitin Pasari, Chetan Rane, et al. Swe-bench pro: Can ai agents solve long-horizon software engineering tasks? *arXiv preprint arXiv:2509.16941*, 2025.

Jingzhe Ding, Shengda Long, Changxin Pu, Huan Zhou, Hongwan Gao, Xiang Gao, Chao He, Yue Hou, Fei Hu, Zhaojian Li, et al. Nl2repo-bench: Towards long-horizon repository generation evaluation of coding agents. *arXiv preprint arXiv:2512.12730*, 2025.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.183. URL https://aclanthology.org/2023.emnlp-main.183/.

Yiran Ding, Li Lyna Zhang, Chengruidong Zhang, Yuanyuan Xu, Ning Shang, Jiahang Xu, Fan Yang, and Mao Yang. Longrope: Extending llm context window beyond 2 million tokens. *arXiv preprint arXiv:2402.13753*, 2024.

Yao Fu, Rameswar Panda, Xinyao Niu, Xiang Yue, Hannaneh Hajishirzi, Yoon Kim, and Hao Peng. Data engineering for scaling language models to 128k context. In *ICML*, volume 235 of *Proceedings of Machine Learning Research*, pp. 14125–14134. PMLR / OpenReview.net, 2024.

Chaochen Gao, Xing Wu, Qi Fu, and Songlin Hu. Quest: Query-centric data synthesis approach for long-context scaling of large language model. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025a. URL https://openreview.net/forum?id=sAYnDWaGd5.

Chaochen Gao, Xing Wu, Zijia Lin, Debing Zhang, and Songlin Hu. Nextlong: Toward effective long-context training without long documents. *arXiv preprint arXiv:2501.12766*, 2025b.

Tianyu Gao, Alexander Wettig, Howard Yen, and Danqi Chen. How to train long-context language models (effectively). In *ACL (1)*, pp. 7376–7399. Association for Computational Linguistics, 2025c.

Aryo Pradipta Gema, Joshua Ong Jun Leang, Giwon Hong, Alessio Devoto, Alberto Carlo Maria Mancino, Rohit Saxena, Xuanli He, Yu Zhao, Xiaotang Du, Mohammad Reza Ghasemi Madani, Claire Barale, Robert McHardy, Joshua Harris, Jean Kaddour, Emile Van Krieken, and Pasquale Minervini. Are we done with MMLU? In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 5069–5096. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.262. URL https://doi.org/10.18653/v1/2025.naacl-long.262.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

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? *CoRR*, abs/2404.06654, 2024.

Liang Hu, Jianpeng Jiao, Jiashuo Liu, Dongyuan Mutu, Yanle Ren, Zhoufutu Wen, Kaiyuan Zhang, Xuanliang Zhang, Xiang Gao, Tianci He, et al. Finsearchcomp: Towards a realistic, expert-level evaluation of financial search and reasoning. In *International Conference on Learning Representations*, volume 2026, pp. 46402–46428, 2026.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025. URL https://openreview.net/forum?id=chfJJYC3iL.

Fengqing Jiang, Yite Wang, Boyi Liu, Zhaoyang Wang, Canwen Xu, Zhewei Yao, Radha Poovendran, and Yuxiong He. Midtool: Mid-training data synthesis for agentic tool use. *CoRR*, abs/2608.20314, 2026.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *ICLR*. OpenReview.net, 2024.

Huayang Li, Pat Verga, Priyanka Sen, Bowen Yang, Vijay Viswanathan, Patrick Lewis, Taro Watanabe, and Yixuan Su. Alr2: A retrieve-then-reason framework for long-context question answering. *arXiv preprint arXiv:2410.03227*, 2024a.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. TACO: Topics in algorithmic COde generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Xiangyi Li, Yimin Liu, Wenbo Chen, Bingran You, Zonglin Di, Yifeng He, Shenghan Zheng, Kyoung Whan Choe, Jiankai Sun, Shuyi Wang, et al. SkillsBench: Benchmarking how well agent skills work across diverse tasks. *arXiv preprint arXiv:2602.12670*, 2026.

Yanyang Li, Shuo Liang, Michael Lyu, and Liwei Wang. Making long-context language models better multi-hop reasoners. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2462–2475, 2024b.

Haoran Lian, Junmin Chen, Wei Huang, Yizhe Xiong, Wenping Hu, Guiguang Ding, Hui Chen, Jianwei Niu, Zijia Lin, Fuzheng Zhang, et al. Breaking the stage barrier: A novel single-stage approach to long context extension for large language models. In *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 4897–4909, 2025.

Jiayu Liu, Qihan Lin, Cheng Qian, Rui Wang, Emre Can Acikgoz, Xiaocheng Yang, Jiateng Liu, Zhenhailong Wang, Xiusi Chen, Heng Ji, et al. Planbench-xl: Evaluating long-horizon planning of llm tool-use agents in large-scale tool ecosystems. *arXiv preprint arXiv:2606.22388*, 2026.

Weiwen Liu, Xu Huang, Xingshan Zeng, Xinlong Hao, Shuai Yu, Dexun Li, Shuai Wang, Weinan Gan, Zhengying Liu, Yuanqing Yu, Zezhong Wang, Yuxian Wang, Wu Ning, Yutai Hou, Bin Wang, Chuhan Wu, Xinzhi Wang, Yong Liu, Yasheng Wang, Duyu Tang, Dandan Tu, Lifeng Shang, Xin Jiang, Ruiming Tang, Defu Lian, Qun Liu, and Enhong Chen. Toolace: Winning the points of LLM function calling. In *ICLR*. OpenReview.net, 2025.

Junru Lu, Jiarui Qin, Lingfeng Qiao, Yinghui Li, Xinyi Dai, Bo Ke, Jianfeng He, Ruizhi Qiao, Di Yin, Xing Sun, Yunsheng Wu, Yinsong Liu, Shuangyin Liu, Mingkong Tang, Haodong Lin, Jiayi Kuang, Fanxu Meng, Xiaojuan Tang, Yunjia Xi, Junjie Huang, Haotong Yang, Zhenyi Shen, Yangning Li, Qianwen Zhang, Yifei Yu, Siyu An, Junnan Dong, Qiufeng Wang, Jie Wang, Keyu Chen, Wei Wen, Taian Guo, Zhifeng Shen, Daohai Yu, Jiahao Li, Ke Li, Zongyi Li, and Xiaoyu

Tan. Youtu-llm: Unlocking the native agentic potential for lightweight large language models, 2026a. URL https://arxiv.org/abs/2512.24618.

Zhiyuan Lu, Chenliang Li, Yingcheng Shi, Weizhou Shen, Ming Yan, and Fei Huang. Corpusqa: A 10 million token benchmark for corpus-level analysis and reasoning. *arXiv preprint arXiv:2601.14952*, 2026b.

Kaijing Ma, Xeron Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. In *International Conference on Learning Representations*, volume 2025, pp. 80062–80161, 2025.

Mike Merrill, Alexander Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Shin, Thomas Walshe, E Kelly Buchanan, et al. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. In *International Conference on Learning Representations*, volume 2026, pp. 40903–40986, 2026.

Niloofar Mireshghallah, Maria Antoniak, Yash More, Yejin Choi, and Golnoosh Farnadi. Trust No Bot: Discovering personal disclosures in human-LLM conversations in the wild. In *First Conference on Language Modeling*, 2024. URL https://maria-antoniak.github.io/resources/2024_colm_trust_no_bot.pdf.

Kaixiang Mo, Yuxin Shi, Weiwei Weng, Zhiqiang Zhou, Shuman Liu, Haibo Zhang, and Anxiang Zeng. Mid-training of large language models: A survey. *arXiv preprint arXiv:2510.06826*, 2025.

NVIDIA. Nemotron-SFT-Agentic-v2. Hugging Face dataset, 2026a. URL https://huggingface.co/datasets/nvidia/Nemotron-SFT-Agentic-v2. Accessed September 12, 2026.

NVIDIA. Nemotron-SFT-SWE-v3.5. Hugging Face dataset, 2026b. URL https://huggingface.co/datasets/nvidia/Nemotron-SFT-SWE-v3.5. Accessed September 12, 2026.

OpenAI. Introducing SWE-bench Verified. Official benchmark release, 2024. URL https://openai.com/index/introducing-swe-bench-verified/. Released August 13, 2024; accessed September 12, 2026.

OpenAI. GraphWalks: A Multi Hop Reasoning Long Context Benchmark. Hugging Face dataset, 2025a. URL https://huggingface.co/datasets/openai/graphwalks. Accessed September 12, 2026.

OpenAI. OpenAI MRCR: Long Context Multiple Needle in a Haystack Benchmark. Hugging Face dataset, 2025b. URL https://huggingface.co/datasets/openai/mrcr. Accessed September 12, 2026.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.

Indraneil Paul, Falko Helm, Goran Glavaš, and Iryna Gurevych. Octolong: Mid-training on cross-repository code contexts enhances long-context modeling. *arXiv preprint arXiv:2608.05141*, 2026.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. In *ICLR*. OpenReview.net, 2024.

Thinh Pham, Nguyen Nguyen, Pratibha Zunjare, Weiyuan Chen, Yu-Min Tseng, and Tu Vu. Sealqa: Raising the bar for reasoning in search-augmented language models. In *International Conference on Learning Representations*, volume 2026, pp. 98989–99017, 2026.

Jiarui Qin, Yunjia Xi, Junjie Huang, Renting Rui, Di Yin, Weiwen Liu, Yong Yu, Weinan Zhang, and Xing Sun. APTBench: Benchmarking agentic potential of base LLMs during pre-training. *arXiv preprint arXiv:2510.24397*, 2025. doi: 10.48550/arXiv.2510.24397. URL https://arxiv.org/abs/2510.24397.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. In *EMNLP*, pp. 2383–2392. The Association for Computational Linguistics, 2016.

Bharat Runwal, Ashish Sunil Agrawal, Anurag Roy, and Rameswar Panda. PRISM: demystifying retention and interaction in mid-training. *CoRR*, abs/2603.17074, 2026.

Ning Shang, Li Lyna Zhang, Siyuan Wang, Gaokai Zhang, Gilsinia Lopez, Fan Yang, Weizhu Chen, and Mao Yang. Longrope2: Near-lossless llm context window scaling. *arXiv preprint arXiv:2502.20082*, 2025.

Rulin Shao, Akari Asai, Shannon Zejiang Shen, Hamish Ivison, Varsha Kishore, Jingming Zhuo, Xinran Zhao, Molly Park, Samuel G. Finlayson, David A. Sontag, Tyler Murray, Sewon Min, Pradeep Dasigi, Luca Soldaini, Faeze Brahman, Wen-tau Yih, Tongshuang Wu, Luke Zettlemoyer, Yoon Kim, Hannaneh Hajishirzi, and Pang Wei Koh. DR tulu: Reinforcement learning with evolving rubrics for deep research. *CoRR*, abs/2511.19399, 2025.

Weizhou Shen, Ziyi Yang, Chenliang Li, Zhiyuan Lu, Miao Peng, Huashan Sun, Yingcheng Shi, Shengyi Liao, Shaopeng Lai, Bo Zhang, Dayiheng Liu, Fei Huang, Jingren Zhou, and Ming Yan. QwenLong-L1.5: Post-training recipe for long-context reasoning and memory management. *CoRR*, abs/2512.12967, 2025. doi: 10.48550/ARXIV.2512.12967. URL https://doi.org/10.48550/arXiv.2512.12967.

Liangcai Su, Zhen Zhang, Guangyu Li, Zhuo Chen, Chenxi Wang, Maojia Song, Xinyu Wang, Kuan Li, Jialong Wu, Xuanzhong Chen, Zile Qiao, Zhongwang Zhang, Huifeng Yin, Shihao Cai, Runnan Fang, Zhengwei Tao, Wenbiao Yin, Chenxiong Qian, Yong Jiang, Pengjun Xie, Fei Huang, and Jingren Zhou. Scaling agents via continual pre-training. *CoRR*, abs/2509.13310, 2025.

Qisheng Su, Zhen Fang, Shiting Huang, Yu Zeng, Yiming Zhao, Kou Shi, Ziao Zhang, Lin Chen, Zehui Chen, Lijun Wu, and Feng Zhao. ACC: compiling agent trajectories for long-context training. *CoRR*, abs/2605.21850, 2026.

Terminal-Bench Team. Terminal-Bench 2.1. Official benchmark release, 2026. URL https://www.tbench.ai/news/terminal-bench-2-1. Accessed September 12, 2026.

Chengying Tu, Xuemiao Zhang, Rongxiang Weng, Rumei Li, Chen Zhang, Yang Bai, Hongfei Yan, Jingang Wang, and Xunliang Cai. A survey on llm mid-training. *arXiv preprint arXiv:2510.23081*, 2025.

Kiran Vodrahalli, Santiago Ontanon, Nilesh Tripuraneni, Kelvin Xu, Sanil Jain, Rakesh Shivanna, Jeffrey Hui, Nishanth Dikkala, Mehran Kazemi, Bahare Fatemi, et al. Michelangelo: Long context evaluations beyond haystacks via latent structure queries. *arXiv preprint arXiv:2409.12640*, 2024.

Siyuan Wang, Gaokai Zhang, Li Lyna Zhang, Ning Shang, Fan Yang, Dongyao Chen, and Mao Yang. Loongrl: Reinforcement learning for advanced reasoning over long contexts. In *International Conference on Learning Representations*, volume 2026, pp. 140078–140097, 2026a.

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. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024.

Zhenting Wang, Qi Chang, Hemani Patel, Shashank Biju, Cheng-En Wu, Quan Liu, Aolin Ding, Alireza Rezazadeh, Ankit Parag Shah, Yujia Bao, et al. Mcp-bench: Benchmarking tool-using llm agents with complex real-world tasks via mcp servers. In *International Conference on Learning Representations*, volume 2026, pp. 97377–97425, 2026b.

Jason Wei, Zhiqing Sun, Spencer Papay, Scott McKinney, Jeffrey Han, Isa Fulford, Hyung Won Chung, Alex Tachard Passos, William Fedus, and Amelia Glaese. Browsecomp: A simple yet challenging benchmark for browsing agents. *arXiv preprint arXiv:2504.12516*, 2025.

Ryan Wong, Jiawei Wang, Junjie Zhao, Li Chen, Yan Gao, Long Zhang, Xuan Zhou, Zuo Wang, Kai Xiang, Ge Zhang, Wenhao Huang, Yang Wang, and Ke Wang. WideSearch: Benchmarking agentic broad info-seeking. *CoRR*, abs/2508.07999, 2025. doi: 10.48550/ARXIV.2508.07999. URL https://doi.org/10.48550/arXiv.2508.07999.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *ICLR*. OpenReview.net, 2025.

Chejian Xu, Wei Ping, Peng Xu, Zihan Liu, Boxin Wang, Mohammad Shoeybi, Bo Li, and Bryan Catanzaro. From 128k to 4m: Efficient training of ultra-long context large language models. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 13122–13133, 2026a.

Peng Xu, Wei Ping, Xianchao Wu, Chejian Xu, Zihan Liu, Mohammad Shoeybi, and Bryan Catanzaro. Chatqa 2: Bridging the gap to proprietary llms in long context and rag capabilities. In *International Conference on Learning Representations*, volume 2025, pp. 67244–67258, 2025.

Xiaoyue Xu, Sikui Zhang, Xiaorong Wang, Xu Han, and Chaojun Xiao. Beyond reward engineering: A data recipe for long-context reinforcement learning. *arXiv preprint arXiv:2606.18831*, 2026b.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025a.

John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. Swe-agent: Agent-computer interfaces enable automated software engineering. In *NeurIPS*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *EMNLP*, pp. 2369–2380. Association for Computational Linguistics, 2018.

Zonghan Yang, Shengjie Wang, Kelin Fu, Wenyang He, Weimin Xiong, Yibo Liu, Yibo Miao, Bofei Gao, Yejie Wang, Yingwei Ma, Yanhao Li, Yue Liu, Zhenxing Hu, Kaitai Zhang, Shuyi Wang, Huarong Chen, Flood Sung, Yang Liu, Yang Gao, Zhilin Yang, and Tianyu Liu. Kimi-dev: Agentless training as skill prior for swe-agents. *CoRR*, abs/2509.23045, 2025b.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *CoRR*, abs/2406.12045, 2024. doi: 10.48550/ARXIV.2406.12045. URL https://doi.org/10.48550/arXiv.2406.12045.

Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. Arctic-embed 2.0: Multilingual retrieval without compromise. *CoRR*, abs/2412.04506, 2024.

Ji Zeng, Dayuan Fu, Tiantian Mi, Yumin Zhuang, Yaxing Huang, Xuefeng Li, Lyumanshan Ye, Muhang Xie, Qishuo Hua, Zhen Huang, Mohan Jiang, Hanning Wang, Jifan Lin, Yang Xiao, Jie Sun, Yunze Wu, and Pengfei Liu. davinci-dev: Agent-native mid-training for software engineering. *CoRR*, abs/2601.18418, 2026.

Liang Zhao, Tianwen Wei, Liang Zeng, Cheng Cheng, Liu Yang, Peng Cheng, Lijie Wang, Chenxia Li, Xuejie Wu, Bo Zhu, et al. Longskywork: A training recipe for efficiently extending context length in large language models. *arXiv preprint arXiv:2406.00605*, 2024.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. Agieval: A human-centric benchmark for evaluating foundation models. In *Findings of the association for computational linguistics: NAACL 2024*, pp. 2299–2314, 2024.

Jiang Zhou, Zhiyuan Fan, Xing Wu, Tinghao Yu, Feng Zhang, and Lilin Wang. Deep research pretraining via predictive navigation. *CoRR*, abs/2608.00432, 2026.

Peilin Zhou, Bruce Leon, Xiang Ying, Can Zhang, Yifan Shao, Qichen Ye, Dading Chong, Zhiling Jin, Chenxuan Xie, Meng Cao, et al. Browsecomp-zh: Benchmarking web browsing ability of large language models in chinese. *arXiv preprint arXiv:2504.19314*, 2025a.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, pp. 15585–15606, 2024.

Yang Zhou, Hongyi Liu, Zhuoming Chen, Yuandong Tian, and Beidi Chen. Gsm-infinite: How do your llms behave over infinitely increasing context length and reasoning complexity? *arXiv preprint arXiv:2502.05252*, 2025b.

Wenhao Zhu, Pinzhen Chen, Hanxu Hu, Shujian Huang, Fei Yuan, Jiajun Chen, and Alexandra Birch. Generalizing from short to long: Effective data synthesis for long-context instruction tuning. *arXiv preprint arXiv:2502.15592*, 2025.

Yonghao Zhuang, Lanxiang Hu, Longfei Yun, Souvik Kundu, Zhengzhong Liu, Eric P Xing, and Hao Zhang. Scaling long context training data by long-distance referrals. In *International Conference on Learning Representations*, volume 2025, pp. 101607–101624, 2025.

A Dataset Details

A.1 Trajectory Sources and Cleaning

Sources and Permissions. Our source corpus consists of agent trajectories generated while carrying out real-world user tasks in software development and workplace settings. Each trajectory records user requests, assistant responses, and tool interactions. All data used in this study were collected with users' knowledge and authorization, with the necessary permissions and licenses obtained for their use in this research.

Privacy Protection. We apply three complementary forms of privacy protection: user-level de-identification, organization-level exclusion, and task-level semantic privacy screening.

User-level de-identification. The replacement pipeline targets three groups of sensitive information:

- **Personal identifiers:** user identifiers, names, email addresses, phone numbers, identity numbers, and payment-card numbers.
- **Authentication material:** API keys, access tokens, embedded passwords, private keys, and database connection strings.
- **Environment-specific identifiers:** personal filesystem paths, version-control identity fields, IP addresses, and internal domains.

Detection combines pattern matching with contextual checks, allowlists, and format or checksum validation where applicable. It operates on message text, including tool observations, as well as reasoning fields and string values in structured tool-call arguments. Detected values are replaced with randomly generated, type-appropriate synthetic values. Repeated values of the same category within a trajectory receive consistent replacements, preserving references across messages and tool interactions.

Organization-level exclusion. We discard entire trajectories containing prominent company identifiers or involving internal documents. The unit of removal is the complete trajectory, including its user requests, assistant responses, and tool observations, so that associated organizational content is excluded together with the identifying material.

Task-level semantic privacy screening. Sensitive information can also be conveyed through task content after explicit identifiers are removed (Mireshghallah et al., 2024). We therefore use an LLM to assess whether a task involves sensitive personal information or confidential company projects. Trajectories flagged by this semantic screening are discarded in full. This criterion addresses privacy and confidentiality in the task's meaning and context, complementing explicit identifier replacement and organization-level filtering.

Task Safety. Separately, an LLM assesses task objectives for harmful intent, including malicious cyberattacks, phishing, fraud, and dangerous weapon-related instructions. It distinguishes such objectives from routine software development and data processing. Trajectories labeled unsafe are excluded in full. Task safety and privacy are separate exclusion criteria: a task with a benign objective may still contain sensitive personal or organizational information.

Cleaning and Filtering. After standardizing the collected trajectories into a unified interaction format, we apply structural and quality checks covering format validity, task completeness, and interaction quality (Table 5). These checks remove malformed, incomplete, and low-value trajectories while retaining informative patterns such as error recovery and progressive exploration.

Domain Classification. We organize the trajectories used in this study into two task domains: *code* and *search*. Assignment combines tool-use patterns with task intent; auxiliary calls such as planning and memory management are excluded from the tool counts.

- **Code.** We identify coding trajectories from the prevalence of code reading, editing, and execution tools, supplemented by agent-scaffold context. A keyword check distinguishes software work from document production. Tasks include feature development, refactoring, bug fixing and debugging, testing, codebase analysis, maintenance automation, configuration management, and DevOps.

Table 5: Trajectory cleaning criteria.

Criterion	Assessment	Selection principle
Format validity	Rules	• Nonempty user request and assistant text or actions. • ≥ 5 non-system messages; ≥ 1 matched tool-call/response pair. • Remove agent-harness boilerplate and tool-output formatting noise.
Task completeness	Rules + LLM	• Remove dangling terminal calls; reject LLM-labeled incomplete workflows. • Allow completed attempts with explicit external constraints.
Quality	Rules	• Reject ≥ 4 consecutive calls with identical tools and arguments. • Reject error-marked responses exceeding 70% of all tool responses.
	LLM	• High-value patterns: error recovery and progressive exploration. • Low-value patterns: unproductive loops, dead ends, and irrelevant actions. • Ratios: summed annotated segment lengths / total nodes; reject low-value ratio $> 50\%$. • Clarify user instructions and refine tool-response content.

- **Search.** We select trajectories with at least two web-retrieval calls accounting for at least 40% of non-auxiliary calls. Retained tasks include information retrieval and question answering, as well as research and report synthesis from retrieved evidence.

A.2 Mid-training Data Statistics

Each mid-training variant combines an approximately 30B-token background corpus with a 10B-token component of original trajectories or TrajLong data. We report lengths of complete training records before packing, using the Qwen3-14B-Base tokenizer without adding BOS or EOS tokens. Repeated sampling occurrences are counted separately; k and B denote 10^3 and 10^9 tokens.

Raw Trajectory Statistics. Table 6 summarizes the code and search trajectory corpora, whose mean lengths are 82.59k and 65.91k tokens, respectively. User turns count user-role messages, and tool calls count individual calls issued by the assistant. Before packing, the data-level loss mask retains supervision on only 22.51% and 9.82% of tokens, respectively; the remaining tokens provide context without direct supervision in the masked variant.

TrajLong Data Statistics. Figure 7 compares the length distributions of original trajectories and the three TrajLong data components using 2k-token bins over 2k–300k, with each curve normalized by its corpus's full record count.

Table 6: Raw code and search trajectory statistics.

Statistic	Code	Search
Training records	127,153	159,017
Total tokens (B)	10.502	10.481
Mean length (k)	82.59	65.91
Mean user turns	4.82	1.94
Mean tool calls	51.9	16.4
Supervised tokens (B)	2.364	1.029
Supervision density (%)	**22.51**	**9.82**

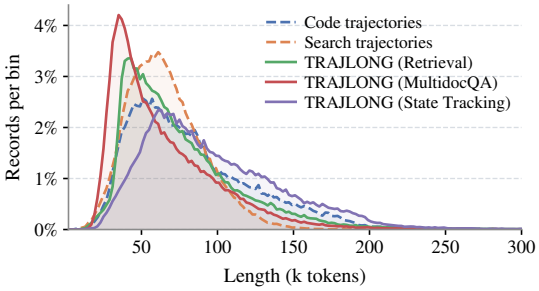


Figure 7: Trajectory and TrajLong length distributions.

Shared Mid-training Corpus. The approximately 30B-token background corpus contains about 5.6 million training records, with a mean length of 5.50k tokens. About 76.2% of records are shorter than 2k tokens. Table 7 reports the mixture composition and token budgets, and Figure 8 visualizes the compiled data alongside the background corpus.

Table 7: Data composition and token budgets of the TrajLong mid-training mixture.

Category	Tokens (B)	Ratio
Code & Repos	9.4	23.50%
Books & Papers	10.1	25.25%
Web	5.1	12.75%
Math & STEM	1.8	4.50%
Instructions & Logs	3.6	9.00%
Compiled Agent Trajectories	10.0	25.00%
Total	**40.0**	**100.00%**

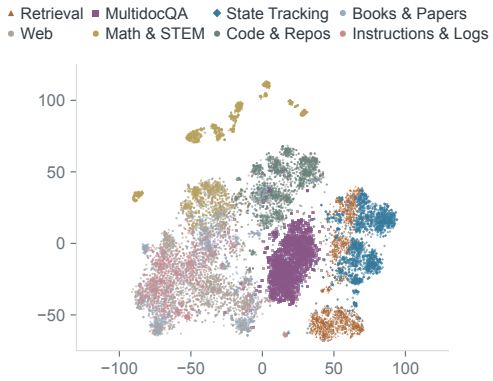


Figure 8: t-SNE visualization of TrajLong data with Arctic-Embed 2.0 (Yu et al., 2024).

A.3 SFT Data

Our SFT mixture comprises five domains from three public datasets. Nemotron-SFT-Agentic-v2 (NVIDIA, 2026a) provides three complementary subsets: *interactive agent* contains multi-turn customer-service conversations involving tool use; *tool calling* covers synthetic single-turn, multi-turn, and multi-step tool-use trajectories; and *search* contains web-search trajectories for answering factual questions.

Nemotron-SFT-SWE-v3.5 (NVIDIA, 2026b) provides software-engineering trajectories collected with OpenCode, covering changes to code, tests, documentation, and configuration files. Following the setting of ProLong (Gao et al., 2025c), for general conversational instruction following, we use the `train_sft` split of UltraChat-200K,[1] a filtered version of UltraChat (Ding et al., 2023).

Table 8: SFT corpus statistics and mixture proportions.

Domain	Records	Tokens (B)	Sampling (%)	Token mix (%)
Nemotron Interactive	278,880	1.220	37.83	40.01
Nemotron Tool Calling	707,052	3.804	36.94	36.86
Nemotron SWE	5,115	0.309	12.47	10.07
UltraChat	207,865	0.247	7.58	8.08
Nemotron Search	5,968	0.152	5.18	4.97
Total	**1,204,880**	**5.732**	**100.00**	**100.00**

Table 8 reports corpus sizes before mixture sampling. Records are complete conversations, and token counts include the Qwen chat template before overlength splitting and packing, without extra BOS/EOS tokens. Sampling weights apply to packed sequences of up to 131,072 tokens. The expected training-token share for domain i is $w_i \bar{L}_i / \sum_j w_j \bar{L}_j$, where w_i is its sampling weight and $\bar{L}_i$ its mean packed length. Supervision covers rendered assistant reasoning, responses, and tool calls, while system messages, user messages, and tool observations are masked.

A.4 Benchmark Decontamination

We screen the raw agent trajectories, generated question–answer (QA) examples, approximately 30B-token background corpus, and downstream SFT data against benchmark test data. Screening combines n-gram matching with semantic matching to identify lexical overlap and semantically equivalent content, including rephrased test material. Both source trajectories and compiled QA examples are checked, covering the original interactions and their derived supervision.

For trajectory-derived data, exclusion operates at the source-trajectory level. If a trajectory or any of its generated QA examples is flagged, we discard the entire trajectory together with all QA examples

[1] Hugging Face H4 UltraChat-200K dataset card.

derived from it, rather than removing only the matched passage or individual QA pair. Matched records in the background corpus and SFT data are likewise excluded from their respective training mixtures.

B Benchmarks and Baselines

This section describes the baseline models, followed by the long-context and agent benchmarks used for downstream SFT and mid-training ICL evaluation.

B.1 Baseline Models

GPT-OSS-120B. OpenAI's gpt-oss-120b (Agarwal et al., 2025) is an open-weight mixture-of-experts reasoning model with approximately 117B total and 5.1B active parameters per token. It supports configurable reasoning effort and native tool calling.

Gemini 2.5 Flash. Gemini 2.5 Flash (Comanici et al., 2025) is Google's multimodal model designed to balance reasoning quality, latency, and computational cost. It supports a controllable thinking budget, long-context input, and tool use; our tables report the thinking configuration.

QwenLong-L1.5-30B-A3B. QwenLong-L1.5 (Shen et al., 2025) builds on Qwen3-30B-A3B-Thinking, combining synthesized long-context reasoning tasks with specialized reinforcement learning. Its reported 256K context window is distinct from the memory-management framework used for longer inputs.

Qwen3-14B and Qwen3-30B-A3B. Both models belong to the Qwen3 family (Yang et al., 2025a) and support thinking and non-thinking modes. Qwen3-14B is a dense model with 14.8B parameters, whereas Qwen3-30B-A3B uses a mixture-of-experts architecture with 30.5B total and 3.3B active parameters. The released models have a native context length of 32,768 tokens; our baselines use inference-time context extension with YaRN (Peng et al., 2024).

ACC (SFT). We use the publicly released ACC dataset (Su et al., 2026) for the baseline in Table 4. We mix ACC with the same original SFT corpus used by TrajLong (SFT) at a 3:7 ratio and fine-tune Qwen3-14B-Base for two effective epochs. The total input-token budget and all remaining SFT settings are identical to those of TrajLong (SFT), as detailed in Appendix D.1.

B.2 Downstream Long-context SFT Benchmarks

For downstream SFT evaluation, we strictly follow the official evaluation protocols of the following six public long-context benchmarks.

LongBench v2. LongBench v2 (Bai et al., 2025) evaluates long-context understanding and reasoning through multiple-choice questions covering single- and multi-document QA, in-context learning, dialogue history, code repositories, and structured data.

MRCR. Multi-round Co-reference Resolution (MRCR), described in Michelangelo (Vodrahalli et al., 2024), requires locating and reproducing a specified response among repeated, similar requests in a long conversation. The OpenAI release (OpenAI, 2025b) includes two-, four-, and eight-needle settings, testing retrieval with order-sensitive disambiguation. We compute scores separately for the 2-, 4-, and 8-needle settings and report their unweighted mean.

GraphWalks. GraphWalks (OpenAI, 2025a) presents a directed graph as an edge list and asks for nodes satisfying a breadth-first-search or parent-retrieval query, testing dependency tracing over long structured inputs. We compute scores separately for the 0–128K and 128–256K context-length intervals and report their unweighted mean.

Oolong-Synth. The synthetic component of Oolong (Bertsch et al., 2025) requires interpreting individual text records and aggregating their properties to answer counting, comparison, and distributional questions across the context. Results are reported in Table 15.

GSM-Infinite (GSM-Inf). GSM-Infinite (Zhou et al., 2025b) generates grade-school arithmetic problems from computational graphs, allowing reasoning complexity and distractor-driven context length to be controlled separately.

CorpusQA. CorpusQA (Lu et al., 2026b) evaluates corpus-level analysis that integrates evidence across many documents, including comparison and statistical aggregation, using synthesized questions with programmatically verifiable answers.

B.3 Downstream Agent SFT Benchmarks

We evaluate downstream SFT models on the following twelve public agent benchmarks, retaining their benchmark-specific scoring rules. The non-binary scores and thresholding conventions used in our tables are specified below; repeated-trial aggregation is detailed in Appendix C.2.

BrowseComp (BC). BrowseComp (Browsing Competition) (Wei et al., 2025) tests persistent web search for hard-to-find facts with short, verifiable answers. Its full evaluation set is denoted BC-full.

BrowseComp-ZH (BC-ZH). BrowseComp-ZH (Zhou et al., 2025a) targets multi-hop information seeking on the Chinese web.

WideSearch. WideSearch (Wong et al., 2025) requires collecting numerous verifiable facts and organizing them into a complete structured answer. We report item-level F1, rather than exact-match accuracy. Avg@3 reports average item-level F1, while BoN@3 reports best-of-three item-level F1 (Item F1 Max@3).

FinSearchComp (FSC-V2). FinSearchComp (Hu et al., 2026) evaluates financial information retrieval and reasoning through time-sensitive data fetching, historical fact lookup, and complex historical investigations.

SealQA. SealQA (Pham et al., 2026) tests fact-seeking under noisy, conflicting, or unhelpful search results.

MCP-Atlas. MCP-Atlas (Bandi et al., 2026) evaluates discovering and composing tools across real Model Context Protocol (MCP) servers to complete multi-step tasks. Gemini 2.5 Pro scores each reference claim as fulfilled, partially fulfilled, or unfulfilled, assigning 1, 0.5, or 0, respectively; their mean gives the response's claim coverage c. Avg@3 reports the mean binary success indicator $\mathbf{1}[c \geq 0.75]$, whereas BoN@3 averages the maximum *unthresholded coverage* over three trials. These columns therefore measure thresholded pass rate and best-of-three coverage, not the mean and maximum of the same trial-level score.

SkillsBench. SkillsBench (Li et al., 2026) evaluates agent performance on diverse tasks paired with reusable procedural Skills and executable verifiers.

WebArena. WebArena (Zhou et al., 2024) evaluates instruction-driven, multi-step interaction with reproducible websites, with success determined by the resulting task state.

Terminal-Bench 2.1 (TB 2.1). Terminal-Bench 2.1 (Merrill et al., 2026; Terminal-Bench Team, 2026) evaluates completing realistic tasks in executable terminal environments. Version 2.1 revises the task specifications, dependencies, and resource settings of Terminal-Bench 2.0.

NL2Repo-Bench (NL2Repo). NL2Repo-Bench (Ding et al., 2025) requires constructing an installable, multi-module Python library from a natural-language specification and an initially empty workspace. Its execution-based score is the fraction of reference tests passed, capped at 1. We

average these normalized per-repository scores for Avg@3 and take their three-trial maximum for BoN@3, rather than requiring every test to pass for a repository to receive credit.

SWE-bench Verified (SWE-Verified). SWE-bench Verified (Jimenez et al., 2024; OpenAI, 2024) is a human-validated subset of SWE-bench: agents modify an existing repository to resolve a GitHub issue, with correctness checked by tests.

SWE-bench Pro (SWE-Pro). SWE-bench Pro (Deng et al., 2025) extends repository-level issue resolution to longer-horizon, complex software-engineering problems across diverse projects.

B.4 Mid-training Long ICL Benchmarks

Long ICL evaluates long-context capabilities directly at mid-training checkpoints, before downstream SFT. Our analyses cover ten tasks from five benchmark families: MRCR, RULER, LongBench v2, GraphWalks, and BrowseCompLong. Table 9 summarizes their tasks, output formats, and evaluation settings. For instruction-oriented benchmarks, we use completion prompts with task-specific demonstrations and answer cues to reduce dependence on chat-style instruction following. The adaptations below specify how the model should continue the prompt while retaining the underlying long-context task.

MRCR-ICL. MRCR (OpenAI, 2025b) presents long conversations containing repeated requests with similar topics and formats. The final query identifies one earlier response by its occurrence order and asks the model to reproduce it with a specified random prefix. This design tests precise retrieval and disambiguation rather than topical matching. We use the 2-, 4-, and 8-needle settings with contexts up to 256K. Scoring first checks the required prefix; a missing or incorrect prefix receives zero. Otherwise, the prefix is removed and the remaining response is compared with the reference using `difflib.SequenceMatcher`.

RULER-ICL. We use six tasks from the RULER (Hsieh et al., 2024) evaluation configuration. *RULER-MultiKey* (`niah_multikey_1`) retrieves the value of a queried key among competing key–value pairs, while *Multivalue* retrieves all values associated with the same key. *QA1* and *QA2* place SQuAD (Rajpurkar et al., 2016) and HotpotQA (Yang et al., 2018) evidence, respectively, among additional documents. *Common Words Extraction* (CWE) requires identifying the 10 most frequent words in a long numbered list, and *Variable Tracking* (VT) requires following assignment chains to identify all variables with a specified value. Their context–query–answer templates provide task-specific answer prefixes, supporting direct continuation by base models.

LongBench v2-ICL. We use the documents, questions, and four answer options from the `truncated_by_qwen_100k` version of LongBench v2 (Bai et al., 2025). We use a two-shot format with synthetic short-document multiple-choice QA demonstrations. Each demonstration gives the selected option; the target prompt ends with `The correct answer is (` to elicit an answer letter. Predictions are evaluated against the reference option. The short, medium, and long subsets contain 180, 215, and 108 questions, respectively; these are source subset labels, not final input-length ranges.

GraphWalks-ICL. GraphWalks (OpenAI, 2025a) requires returning the nodes at exactly a specified breadth-first-search depth or the direct parents of a queried node in a directed graph. We prepend four independent demonstrations: BFS at depths one and two, and parent queries with nonempty and empty answers. Each demonstration contains its own graph, operation, and answer, avoiding a pattern that encourages the model to generate another operation instead of answering. The test graph and operation follow, ending with `Final Answer: [` so generation continues inside the answer list. We remove test instances with empty gold node sets to prevent credit from trivial empty-list responses. The reported evaluation uses 637 questions across six length groups labeled 4K–128K.

BrowseCompLong-ICL. BrowseCompLong[2] adapts BrowseComp (Wei et al., 2025) to multi-document QA, requiring evidence aggregation across supplied webpage documents without live

[2]https://huggingface.co/datasets/openai/BrowseCompLongContext

Table 9: Long ICL benchmark overview. MCQ denotes multiple-choice question answering; TC denotes text completion. Length denotes nominal settings or subset labels, except for BrowseCompLong-ICL, which reports the observed complete-prompt token range; S/M/L denotes short, medium, and long subsets. Size counts evaluated queries, excluding ICL demonstrations. MRCR includes the 2-, 4-, and 8-needle settings. RULER covers six selected subtasks, each evaluated at 4K, 8K, 16K, 32K, 64K, and 128K. BrowseCompLong-ICL uses three randomly selected demonstrations; its complete prompts span approximately 20K–243K tokens in our evaluation.

Benchmark	**Task**	**Description**	**Format**	**Length setting**	**Size**
MRCR-ICL	Co-reference resolution	Reproduce an ordinally specified response among repeated requests, with the required prefix.	TC	$\leq$256K	1,711
RULER-ICL	RULER-MultiKey	Retrieve the queried key's value among key–value pairs and distractors.	TC	4K–128K	300
	Multivalue	Retrieve all values associated with the queried key.	TC	4K–128K	300
	QA1	Answer a SQuAD question using a relevant passage among distractors.	TC	4K–128K	300
	QA2	Combine supporting documents to answer a HotpotQA question.	TC	4K–128K	300
	CWE	Identify the 10 most frequent words in a long numbered list.	TC	4K–128K	300
	VT	Follow assignment chains to return all variables sharing a queried value.	TC	4K–128K	350
LongBench v2-ICL	Long-context reasoning	Select the correct answer by reasoning over long documents, dialogue, code, or structured data.	MCQ	S/M/L	503
GraphWalks-ICL	Directed graph queries	Return nodes at exactly a specified BFS depth or direct parents of a node in a directed graph.	TC	4K–128K	637
BrowseCompLong-ICL	Multi-document QA	Answer multi-hop questions by integrating supplied webpage evidence without external browsing.	TC	20K–243K	310

browsing. Each prompt includes three randomly selected, answered demonstrations followed by a fourth, unanswered query. The 32K label refers to the base-dataset setting; complete evaluation prompts, including demonstrations, span 20,244–243,265 tokens. The model generates a short text answer, evaluated using `ConsistencyEval`. Responses are instructed to end with `Finished`, separated by two newlines on each side; when this delimiter is present, only the preceding text is retained for scoring.

Score aggregation. We average scores equally across the three MRCR needle settings, the six GraphWalks length groups, and the three LongBench v2 length subsets. For all six RULER tasks (RULER-MultiKey, Multivalue, QA1, QA2, CWE, and VT), reported scores equally average 4K, 8K, 16K, 32K, 64K, and 128K settings. The GraphWalks + LongBench v2 learning-dynamics proxy gives equal weight to the two benchmark means rather than pooling their individual settings. Gains use the same settings as their reference model, and are reported in percentage points.

B.5 Mid-training Agent ICL Benchmarks

Following APTBench (Qin et al., 2025), we formulate agent tasks as multiple-choice or text-completion probes to assess base models' agent atomic capabilities before SFT. We adapt and extend this design with line-level bug localization and dedicated probes for code-summary selection, failure-aware multi-hop replanning, tool selection, and argument filling.

Agent ICL evaluates individual agent capabilities at mid-training checkpoints, before downstream SFT. We use seven tasks derived from public datasets, including SWE-bench Verified and Lite (Jimenez et al., 2024; OpenAI, 2024), BFCL (Patil et al., 2025), TACO (Li et al., 2023), PlanBench-XL (Liu et al., 2026), and MCP-Bench (Wang et al., 2026b). They cover code understanding and repair, planning and replanning, and tool use, complementing end-to-end agent evaluations. Task-specific completion prompts elicit a line number or an answer choice from the supplied task context, with demonstrations where applicable. All tasks report accuracy. Table 10 summarizes their objectives, output formats, and sizes.

Table 10: Agent ICL benchmark overview. MCQ denotes multiple-choice question answering; TC denotes text completion.

Benchmark	Description	Format	Size
Bug Localization	Locate the faulty line using an issue description and numbered code.	TC	114
Bug Fixing	Select the correct replacement for a specified faulty line.	MCQ	114
Code Summary	Identify the summary that matches a code snippet's behavior.	MCQ	235
Planning Preference	Choose the next action from the task goal and interaction history.	MCQ	181
Multi-hop Replanning	Select a feasible continuation or task status after tool failure.	MCQ	210
Tool Selection	Match a user request to a tool using candidate descriptions.	MCQ	103
Argument Filling	Select a missing argument value using the interaction history.	MCQ	325

Bug Localization. This task connects issue understanding with fault localization: the model identifies the line responsible for a reported defect in a numbered source file. Three labeled demonstrations establish the completion format, and the test prompt ends with `The buggy line number is:` to elicit a line number.

Bug Fixing. With the faulty line identified, the model must distinguish a correct repair from three alternative replacements using the issue description and surrounding code. Three demonstrations

illustrate the four-choice format. The prompt ends with `Answer:`, where the four candidate letters are scored.

Code Summary. Code Summary tests whether a model understands a snippet's functionality and behavior by choosing among four candidate summaries. Differences in conditions, operation order, or returned quantities require attention to program semantics. The model completes an `Answer:` cue with the selected letter.

Planning Preference. The user query and preceding search or browsing history provide the evidence for choosing the next action from six candidates. This tests whether a model can identify productive follow-up actions and recognize when enough information has been gathered to conclude the task. Three demonstrations precede the test question, which ends with `Your choice:`.

Multi-hop Replanning. Tool unavailability can invalidate a previously feasible plan. Using tool descriptions, execution history, and failure feedback, the model chooses among four alternatives representing a continuation plan, a next action, or a task-status judgment. These questions probe dependency tracking and adaptation to disrupted execution paths.

Tool Selection. The model selects a tool from three or four candidates for a user request by interpreting the supplied function descriptions. Closely related tool functions and differing constraints test semantic discrimination beyond name matching. Three demonstrations illustrate the task before the model completes the test question's `Answer:` cue.

Argument Filling. This task focuses on binding information from an interaction history to the next tool call. A missing argument is marked in the call, and the model chooses its value from four candidates using the tool descriptions and previous observations. Correct choices require resolving entities and their roles in context.

Scoring and aggregation. Each question yields one scored prediction. Bug Localization, Code Summary, Planning Preference, and Tool Selection use greedy completion at temperature zero, followed by extraction of the predicted line number or answer letter. Bug Fixing, Multi-hop Replanning, and Argument Filling instead select the candidate letter with the highest final-token log probability at the answer position. Predictions are checked against the reference answers. Multi-hop Replanning uses overall question accuracy rather than a macro-average across its question categories.

Bug Localization and Bug Fixing each contain 82 Python and 32 Multilingual questions; we give the two subset accuracies equal weight. The Tool Selection + Argument Filling proxy in the learning-dynamics analysis likewise averages the two task scores equally. Reported gains use the reference model specified in each figure and are expressed in percentage points.

C Evaluation Protocols

C.1 ICL Evaluation

Inference Settings. We evaluate mid-training checkpoints before SFT using Long ICL, Agent ICL, and general ICL, described in Appendices B.4, B.5, and E.3. Agent ICL probes individual atomic capabilities from supplied contexts, without executing an interactive agent scaffold. We use a completion interface with temperature 0 and a default generation limit of 1,024 tokens. The serving context limits are 262,144 tokens for Long and Agent ICL and 65,536 tokens for general ICL. All ICL and SFT evaluations set `truncation_side=right`. No separate thinking budget is specified for ICL.

Long ICL. For Long ICL, we generate one greedy completion per question, with the task-specific scoring and subset aggregation described in Appendix B.4. Generation is capped at 1,024 tokens for MRCR, RULER, GraphWalks, LongBench v2, and BrowseCompLong. LongBench v2 additionally stops at a closing parenthesis or newline. We specify no custom stop sequence for the other four tasks.

Agent ICL. Agent ICL uses one trial per question. Bug Localization, Code Summary, Planning Preference, and Tool Selection use greedy completion; Bug Fixing, Replanning, and Argument Filling instead rank answer candidates using terminal-token log-probabilities. For these likelihood-based probes, requests with `max_tokens=1` obtain prompt log-probabilities; candidate-wise requests are not repeated sampled trials.

General ICL. General ICL similarly retains benchmark-specific scoring: PIQA scores candidate continuations, MMMLU-Lite uses terminal-token log-probabilities, and AGIEval-CN generates an answer with a newline stop condition.

C.2 SFT Evaluation

Generation Settings. After SFT, we evaluate long-context question answering and interactive agent tasks separately from the ICL probes. For Qwen3-14B, we use task-level defaults of temperature 0.7, top-p 1.0, no top-k cutoff, and a repetition penalty of 1.0. The serving context window is 262,144 tokens, with a default output cap of 65,536 tokens per model call. We specify no custom stop sequence at the task level. Results are reported using Avg@3 and BoN@3, as defined below.

Coding and Tool-use Agents. We execute coding and tool-use agents with benchmark-specific scaffolds. For SWE-bench Verified and SWE-Pro, we use `swe_agent` (Yang et al., 2024) with at most 250 iterations and bounded retries for tool-error recovery. SWE-bench Verified additionally uses a 131,072-token context limit. For Terminal-Bench, we use `terminus2` with XML action parsing, at most 500 iterations, context summarization, and a 229,376-token context limit. NL2Repo uses the Claude Code scaffold with at most 250 turns. MCP-Atlas uses an MCP-enabled agent with at most 100 iterations. Agent execution and benchmark evaluation are subject to task-specific timeouts.

Search Agents. We use the same agent execution settings across all search benchmarks: web search, webpage fetching, and a Python interpreter, with at most 200 iterations and history summarization. Bounded retries handle failed model calls, invalid tool arguments, and tool-call errors. Retrieved webpage content is length-limited and progressively truncated when needed to fit the context budget.

Scoring and Repeated Trials. We retain benchmark-specific scoring: CorpusQA uses GPT-OSS-120B, and SWE-bench Verified uses patch execution with `FAIL_TO_PASS` and `PASS_TO_PASS` tests. Agent score definitions appear in Appendix B.3. In particular, MCP-Atlas Avg@3 averages thresholded success indicators, whereas its BoN@3 uses continuous claim coverage; WideSearch and NL2Repo retain graded scores in both Avg@3 and BoN@3. Failed trials receive zero and remain in the denominator, including timeouts, unrecovered execution or tool errors, invalid answers, and invalid scoring records.

We report average and best-of-N performance at $N = 3$ (Avg@3 and BoN@3), averaged across examples using the benchmark-specific scores above. For MRCR and GraphWalks, this averaging is performed within each subset before the equal-weight aggregation specified in Appendix B.2. Binary scores measure task success, while graded scores retain partial credit.

D Implementation Details

D.1 Training Details

We adopt the open-source ProLong framework (Gao et al., 2025c) for mid-training and downstream supervised fine-tuning. Tables 11 and 12 report the mid-training and downstream SFT settings for Qwen3-14B and Qwen3-30B-A3B under a 256-GPU reference configuration. Both stages retain 262,144 position embeddings and use boundary-preserving packing. Mid-training supervises all valid prediction positions; SFT supervises assistant reasoning, responses, and tool calls with running-mean valid-token normalization.

Mixture Sampling and Token Accounting. Mixture proportions are defined by input-token counts. For the All variant, Retrieval, MultidocQA, and State Tracking contribute 3B, 4B, and 3B

tokens, respectively. Each single-capability variant uses its own 10B-token corpus without repeated sampling. All token budgets refer to actual input tokens processed during training.

Table 11: Mid-training configuration. Global batch size counts independent packed sequences, excluding sequence-parallel replicas.

Parameter	Configuration
Maximum sequence length	262,144 tokens
RoPE base	16,000,000 (no additional scaling)
Optimizer	AdamW
Adam β_1, β_2; ϵ	$(0.9, 0.95)$; 10^{-8}
Peak / minimum learning rate	4×10^{-5} / 4×10^{-6}
Learning-rate schedule / warmup	Cosine / 10%
Weight decay / gradient clipping	0.1 / 1.0
Global / per-device microbatch size	64 / 1
Reference GPU count	256
Sequence-parallel degree	8
Gradient accumulation	2
Distributed training	FSDP full sharding + sequence parallelism
Precision / random seed	BF16 mixed precision / 42
Gradient checkpointing	Enabled
Checkpoint interval	Every 300 steps and the final step

Table 12: Downstream SFT configuration. The positional configuration is retained from mid-training, while training sequences are capped at 128K tokens.

Parameter	Configuration
Maximum sequence length	131,072 tokens
RoPE base	16,000,000 (no additional scaling)
Optimizer	AdamW
Adam β_1, β_2; ϵ	$(0.9, 0.95)$; 10^{-8}
Peak / minimum learning rate	2×10^{-5} / 2×10^{-6}
Learning-rate schedule / warmup	Cosine / 3%
Weight decay / gradient clipping	0.0 / 1.0
Global / per-device microbatch size	32 / 1
Reference GPU count	256
Sequence-parallel degree	8
Gradient accumulation	1
Distributed training	FSDP full sharding + sequence parallelism
Precision / random seed	BF16 mixed precision / 42
Gradient checkpointing / attention	Enabled / FlashAttention-3
Logit block size	2,048

D.2 Retrieval Data Construction

Retrieval converts recorded search passages and file contents into exact-reproduction questions over the complete preprocessed trajectory. We extract candidate items, construct questions that distinguish similar observations or repeated file interactions, align reference answers with the rendered context, and obtain and filter teacher-generated solutions.

Evidence Extraction and Processing. We preserve the original interaction order and associate each candidate with its trajectory position and source metadata. Candidate extraction differs between search and code trajectories:

- **Search passages.** We unpack structured search responses into individual passages, accommodating passage, snippet, and content fields across tool formats. Results from the same search response

form a candidate group in their original order. We retain the associated query, result titles, and passage positions. These groups provide topically related candidates without inserting synthetic distractors.

- **File contents.** Read targets are file contents returned in tool observations, whereas write targets are content payloads supplied in tool-call arguments, rather than write acknowledgments. We normalize tool names, recover readable text from structured payloads, and remove system reminders. We match calls and responses only when their correspondence is unambiguous, recording the file path, operation type, and occurrence for each candidate.

Question Construction. For search trajectories, an LLM summarizes each group's query and result titles into a short topic phrase, falling back to the original query when needed. Eligible groups contain at least five passages; target passages must contain at least 100 characters and occur only once among the trajectory's extracted search passages. We deduplicate candidate groups by topic and sample one eligible target per selected group. When more than 12 groups remain, we retain the 12 with the largest total eligible passage lengths. Questions specify the topic and the target's original one-indexed position within its group. Seventy percent of search questions additionally require a random 10-character alphanumeric prefix before the reproduced passage, while the remaining 30% request the passage alone. We do not renumber passages after target filtering or add extra questions from an already selected group to fill the question budget.

For code trajectories, questions specify the file path, read/write operation, and occurrence, distinguishing repeated interactions with the same file. We sample read and write targets at a target ratio of 7:3, filling shortages from the other type. Requests may additionally require a random prefix before the reproduced content; 30% of code questions use this constraint. Both prefixed and plain requests ask for the recorded content, not a summary or a reconstructed file state.

Context Rendering and Reference Alignment. We render the complete preprocessed trajectory without further truncating tool observations. Each search passage receives its group's topic header, while file blocks retain explicit read/write cues. Reference answers are recovered from the rendered target, rather than copied from an earlier raw representation, so that formatting cleanup and consistent de-identification do not create answer–context mismatches. We require the rendered target to occur exactly once in the context and discard questions with missing or ambiguous alignments. Other trajectory content remains available as context and distractors.

Solution Generation and Filtering. We use DeepSeek-v4-pro (DeepSeek-AI, 2026) as the teacher model to solve retrieval questions given the rendered trajectory. We compare the final answer extracted from each solution with the reference answer, checking required prefixes and applying the same fidelity criteria to search and code questions. Exact matches and matches under Unicode and whitespace normalization are accepted directly. Otherwise, solutions are retained only when their final answers achieve a `SequenceMatcher` similarity of at least 0.95 with the reference. Lower-scoring solutions and failed generation requests are discarded. We append accepted question–solution pairs after the shared trajectory context and remove records with no surviving questions. Table 17 gives an example.

D.3 MultidocQA Data Construction

MultidocQA converts the preprocessed trajectories into questions that require combining evidence across sources. We first extract and organize the evidence, then construct and refine questions, and finally obtain teacher-generated solutions using the complete evidence collection.

Evidence Extraction and Processing. We split each trajectory into interaction turns initiated by user messages. GPT-OSS-120B groups these turns according to continuity of the task, the objects being manipulated, and the resulting artifacts. Turns that develop the same feature or research deliverable, including follow-up requests and related debugging or verification, belong to the same group. A group may contain nonconsecutive turns when the user returns to an earlier task. We check that the groups cover all turns without duplication.

Within each group, we extract file, document, and execution evidence from tool interactions. We check content validity and exclude empty or uninformative responses, inaccessible or removed con-

tent, binary data, visibly corrupted text, and fragments below minimum length thresholds. We then process each evidence type as follows:

- **File evidence.** We extract code and configuration contents from file reads and write/edit operations. We remove system reminders, terminal escape sequences, and non-printing control characters, retaining the recorded code/configuration text and edit fragments.
- **Document evidence.** We collect retrieved passages, fetched webpage text, and textual documents read or produced during the task. We unpack structured responses to recover titles and text, removing tool wrappers, image markup, and URL or metadata clutter.
- **Execution evidence.** We extract standard output and error text from command results, remove runtime-injected noise, and filter failure outputs without useful content. Previously observed contents of referenced scripts or files supplement the results when available.

We retain groups with at least four distinct source keys, based on file paths, commands, or document content; versions of the same file count as one source. Each item receives a unique identifier and source metadata. For each question q_j, the generator selects substantive, interpretable supporting evidence $S_j \subseteq E(\tau)$ from the retained evidence collection $E(\tau)$.

Version Organization and Context Construction. Observed contents of the same file share a source identity, with the version label advancing when the content changes. To construct the long context, we group evidence by source, order available versions within each group, and randomly permute the source groups. Each block displays its evidence identifier, source title, and version label when available. Temporal questions use explicit version or time cues rather than block positions.

Question Construction. The process comprises three stages: diverse question generation, enrichment of evidence dependencies, and final refinement and checking.

Stage 1: Diverse question generation. Gemini-3.5-Flash generates questions, reference answers, and supporting evidence identifiers from each annotated task group. We use nine question categories and nine answer categories (Table 13). For each group, we sample two question categories uniformly and two answer categories from the union of compatible answer categories using category-specific weights; both samplings are without replacement. Answer sampling favors lists, multi-point analyses, and constrained outputs. Each question must use a compatible category pairing, be answerable from the evidence independently of the agent dialogue, and omit evidence identifiers. We validate supporting identifiers and require at least two distinct sources.

Stage 2: Enriching multi-hop dependencies. We strengthen cross-source dependencies through up to two refinement rounds. Each round supplies the current supporting set and up to ten additional candidates, combining relevance ranking by shared entities, version relations, and tool-interaction proximity with random selection. The model is prompted to extend the question by linking facts into a multi-hop chain or combining complementary constraints. Added evidence should contribute to the solution, while the question category and answer requirements remain unchanged.

Stage 3: Final refinement and checking. The model prunes redundant supporting items from S_j, while retaining $E(\tau)$ for the long context, and reviews answer support, ambiguity, and distractor validity. It clarifies wording and output requirements and removes solution steps or explicit retrieval routes from questions. Rule-based checks validate evidence identifiers, category compatibility, and option–answer consistency; semantic concerns are recorded for subsequent solution checks.

Solution Generation. For each question, DeepSeek-v4-pro receives the complete retained evidence collection $E(\tau)$ and that question, without its reference answer or annotated supporting set. We encourage it to use evidence identifiers to track relevant information while solving the question. All questions from a trajectory share the same evidence order and are solved independently. After checking final-answer agreement and solution consistency, we combine accepted questions into a shared-context training record. For each question, the training response combines the teacher-generated solution steps with the reference answer. Table 18 gives an example.

Table 13: Question and answer categories used to construct MultidocQA data. Question categories guide the reasoning task, while answer categories specify response requirements. Only compatible combinations are used.

Question category	**Reasoning requirement**
Cross-document or version comparison	Identify shared conclusions, differences, or changes across related documents or versions.
Multi-hop fact composition	Use information from one source to locate and interpret the next source in a chain.
Quantitative reasoning	Combine numerical facts to compute a derived quantity with a meaningful interpretation.
Aggregation	Count, rank, deduplicate, or identify extrema across related records.
Relational reasoning	Connect evidence through calls, data flow, or relationships between definitions and their use.
Conditional reasoning	Infer an outcome under stated conditions or a hypothetical change.
Global synthesis	Integrate the main conclusions, positions, or design rationale across sources.
Temporal or causal ordering	Establish event order or causal relations using explicit cues in the evidence.
Fine-grained discrimination	Distinguish supported or unsupported statements among closely related alternatives.
Answer category	**Output requirement**
One-sentence answer	A single natural-language sentence stating the requested conclusion.
Entity	A specific name, identifier, path, term, or other short factual item.
Numerical answer	A numerical value with specified units or precision.
Single choice	The label of one selected option.
Multiple choice	The labels of all selected options.
List	A collection or sequence of items, with ordering or fields specified when needed.
Multi-point analysis	An explanation covering the requested aspects of the answer.
Constrained output	An answer satisfying explicit structural or formatting requirements, such as a specified JSON schema.
Verbatim quotation	Exact reproduction of a requested evidence span.

D.4 State Tracking Data Construction

State Tracking converts coding-agent trajectories into questions about evolving file states and the origins of their contents. We reconstruct per-file histories, derive reference answers programmatically, and obtain teacher-generated solutions for two question families: *Snapshot* and *Provenance*.

Evidence Extraction and Processing. We deduplicate trajectories within each agent harness using normalized user requests and tool/file-event sequences, retaining more complete representatives when one trajectory is a prefix of another. Tool calls are aligned with their responses using call identifiers. We normalize file paths and operation names and retain message positions, operation arguments, and execution outcomes. Events are then organized into per-file timelines. Only explicitly successful operations contribute to state updates; failed calls are not counted as successful modifications, and uncertain execution outcomes are not treated as confirmed updates.

State Reconstruction and Verification. A reconstruction segment begins from a complete Write payload or a Read whose completeness can be established from its arguments and returned contents. Partial or truncated reads are not treated as complete initial states. We replay successful edits in order, checking that each replacement can be applied unambiguously and preserving untouched content. A Write replaces the complete file state. When available, subsequent complete reads are compared with the reconstructed state. We distinguish read-verified histories from deterministic histories supported only by complete Write or strict Read anchors, rather than requiring a later read for every sample. Unreplayable edits, uncertain execution outcomes, ambiguous same-file ordering, or inconsistent observations break the segment. Reconstruction can resume from a new complete anchor. Potential unobserved changes from shell commands or sub-agents are recorded as separate risk indicators.

Question Construction. The two families share the reconstructed histories but query different aspects of state evolution. Table 14 summarizes their seven question subtypes and corresponding answer formats:

- **Snapshot.** Questions request the complete file immediately before or after a specified successful Edit, at the end of the retained history, or after rolling back a suffix of consecutive edits. Reference answers are the corresponding reconstructed snapshots. Rollback is restricted to the supported edit suffix rather than arbitrary deletion of an earlier operation. These questions require composing partial updates while preserving the rest of the file, not reproducing an individual edit payload.
- **Provenance.** Questions identify the last operation that wrote a surviving fragment, the operation after which it first appeared, or the operation that removed it. During replay, content spans retain their source labels unless overwritten, while a full Write resets their provenance. Candidate fragments must have an unambiguous answer. Removal questions additionally require the fragment to exist immediately before the selected operation, disappear immediately afterward, and remain absent from all subsequent known states. Answers specify an operation type and its successful-operation index within the target file. Edit, Write, and Delete are counted separately from one, while an initial anchor uses index zero.

We diversify question wording using Chinese and English template pools and LLM-based paraphrasing, while keeping the target file, temporal boundary, operation-counting convention, and output requirements explicit and unchanged.

Context Construction and QA Organization. The rendered trajectory provides the context, and constructed questions are kept separate from the recorded agent dialogue. For Snapshot questions, the context ends before the first subsequent successful Read of the target file, including partial reads, to prevent direct copying of the requested state. If no such read exists, we retain the complete trajectory. Questions can share a training record only when their rendered contexts agree. Among Snapshot questions targeting the same state, we retain one formulation, with weighted sampling favoring final-state questions. Within each record, Snapshot questions follow the temporal order of their target states, and Provenance questions precede Snapshot questions about the same file. Before serialization, we additionally check cross-QA compatibility under this ordering. A QA combination is retained only if later answers cannot be directly inferred from complete file states supplied by preceding Snapshot answers, including for questions concerning the same file.

Table 14: Question subtypes and answer formats used to construct State Tracking data.

Snapshot subtype	**Question target**
After edit	Recover the complete file immediately after its K-th successful Edit.
Before edit	Recover the complete file immediately before its K-th successful Edit, incorporating preceding updates.
Final state	Recover the complete file at the end of its reconstructible state history.
Suffix rollback	Recover the file after undoing the last R successful Edits in a consecutive suffix, without crossing a Write or Delete boundary.
Provenance subtype	**Question target**
Final-state last writer	Identify the Edit or Write that last wrote a target fragment still present in the final state.
First introduction	Identify the earliest operation after which a target fragment appeared, or the initial anchor if it was already present.
Removal	Identify the Edit, Write, or Delete that removes a previously present fragment, after which it remains absent from all subsequent known states.
Question family	**Answer format**
Snapshot	The complete target file contents, preserving unchanged text rather than returning an edit fragment or summary.
Provenance	A structured answer specifying the operation type and its successful-operation index within the target file. Indices are counted separately by operation type from one; initial-anchor answers use zero when applicable.

Difficulty Characterization. We characterize difficulty at the question level using successful target-file mutations before the dependency boundary, the queried operation index, context length, and interference from other files. Snapshot questions additionally track rollback depth and answer length; Provenance questions track mutations after the source event. Output length is treated separately from the complexity of reconstructing the state. Reconstruction confidence and hidden-mutation risk are quality attributes rather than difficulty labels.

Solution Generation and Filtering. We use DeepSeek-v4-pro as the teacher model to solve both Snapshot and Provenance questions using the rendered trajectory context. Snapshot final answers must match the reference under normalization and achieve a character-similarity score of at least 0.99. Provenance final answers must match the structured reference answer. We retain accepted teacher-generated solutions, including their original final answers. Each training record contains the shared trajectory context once, followed by the accepted question–solution pairs in their prescribed order. Table 19 gives an example.

E Additional Experiments and Analysis

E.1 Oolong-Synth Results

Table 15 reports supplementary long-context information aggregation results under the same evaluation protocol as Table 1.

Table 15: SFT performance on Oolong-Synth. Subscripts show gains over the corresponding Qwen3 baseline (pp). The best scores for each backbone LLM are **bolded**.

Models	Oolong-Synth	
	Avg@3	BoN@3
Frontier Models		
Gemini-2.5-Flash-Thinking	78.17	91.00
QwenLong L1.5-30B-A3B	75.50	89.00
GPT-OSS-120B	68.00	84.50
Our Models		
Qwen3-14B	52.17$_{+0.0}$	70.00$_{+0.0}$
- SFT *(w/o Mid-training)*	53.17$_{+1.0}$	71.00$_{+1.0}$
- TrajLong *(Retrieval)*	57.17$_{+5.0}$	77.00$_{+7.0}$
- TrajLong *(MultidocQA)*	57.33$_{+5.2}$	**79.00**$_{+9.0}$
- TrajLong *(State Tracking)*	56.00$_{+3.8}$	75.50$_{+5.5}$
TrajLong-14B-All	**58.50**$_{+6.3}$	78.00$_{+8.0}$
Qwen3-30B-A3B	41.33$_{+0.0}$	52.50$_{+0.0}$
- SFT *(w/o Mid-training)*	52.33$_{+11.0}$	**75.50**$_{+23.0}$
TrajLong-30B-A3B-All	**53.83**$_{+12.5}$	72.50$_{+20.0}$

E.2 Additional Agent Evaluation Results

Table 16: Overall SFT performance (BoN@3, %) across diverse search, tool-use, and coding agentic benchmarks. BC-ZH, FSC-V2, and TB 2.1 denote BrowseComp-ZH, FinSearchComp V2, and Terminal-Bench 2.1, respectively. MCP-Atlas reports unthresholded claim coverage; scoring and three-trial aggregation are defined in Appendices B.3 and C.2. Empty cells indicate unavailable or invalid evaluation results.

Models	Search					Tool Use			Coding			
	BC	BC-ZH	WideSearch	FSC-V2	SealQA	MCP-Atlas	SkillsBench	WebArena	TB 2.1	NL2Repo	SWE-Verified	SWE-Pro
Frontier Models												
Gemini-2.5-Flash-Thinking	12.72	43.42	44.09	46.44	38.75	47.47	16.67	51.98	34.83	12.40	48.60	24.08
QwenLong L1.5-30B-A3B	6.48	21.35	45.14	32.51	28.02	62.77	4.17	36.12	4.49	15.09	43.60	19.56
GPT-OSS-120B	37.91	38.41	51.01	55.75	34.23	69.83	-	43.23	22.47	6.88	67.00	40.63
Our Models												
Qwen3-14B	3.79	19.65	41.40	32.74	25.32	38.82	0.83	31.00	1.12	7.16	29.00	8.62
- SFT *(w/o Mid-training)*	22.99	32.84	43.03	48.52	44.41	74.99	12.50	40.39	15.73	11.94	56.60	37.62
- TrajLong *(Retrieval)*	23.93	35.95	47.55	57.01	42.07	76.75	23.61	42.84	22.47	29.82	66.40	46.24
- TrajLong *(MultidocQA)*	25.28	32.91	47.41	57.34	41.89	77.46	16.67	42.75	19.10	26.91	66.00	48.29
- TrajLong *(State Tracking)*	22.91	37.13	50.56	56.09	45.86	77.19	29.17	42.06	26.97	29.28	67.20	48.97
TrajLong-14B-All	22.83	35.26	47.85	57.72	41.35	75.62	20.83	42.03	24.72	26.08	67.60	49.66
Qwen3-30B-A3B	4.42	16.16	34.46	26.24	24.68	50.75	2.78	33.12	4.49	13.23	35.00	15.73
- SFT *(w/o Mid-training)*	19.91	31.90	38.07	51.99	44.23	72.01	12.50	38.20	5.62	18.29	51.40	37.76
TrajLong-30B-A3B-All	20.85	36.57	46.12	56.29	42.88	72.24	18.06	38.60	21.35	26.49	64.80	47.33

E.3 General ICL Evaluation after Mid-training

We evaluate general capabilities before SFT, comparing Qwen3-14B-Base at initialization, Combined Trajectories, and four TrajLong variants: Retrieval, MultidocQA, State Tracking, and All. We report each trained model's final checkpoint on ten the ten general-capability benchmarks described below under their respective ICL settings.

LiveCodeBench (LCB) (Jain et al., 2025) tests code generation on programming-contest problems, while the English and Chinese versions of FullStackBench (FSB-EN and FSB-ZH) (Cheng et al., 2024) cover practical programming tasks across application domains and programming lan-

guages. MMLU (Hendrycks et al., 2020) assesses knowledge and problem solving across 57 subjects; MMLU-Redux (Gema et al., 2025) improves annotation quality through manual review, and MMLU-Pro (Wang et al., 2024) introduces harder, reasoning-focused questions with up to ten answer options. AGIEval-CN (Zhong et al., 2024) evaluates understanding and reasoning on Chinese examination questions. KOR-Bench (Ma et al., 2025) tests reasoning with newly specified rules while reducing reliance on prior domain knowledge, and PIQA (Bisk et al., 2020) assesses physical commonsense in everyday tasks. Finally, the public MMMLU-Lite subset[3] evaluates multilingual knowledge understanding using translated MMLU questions.

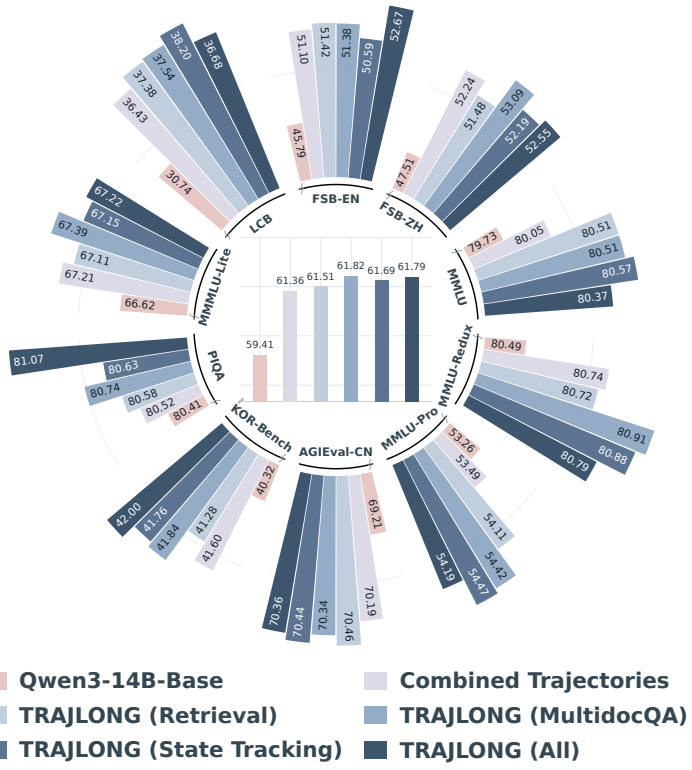


Figure 9: General ICL performance after mid-training on ten evaluations. Outer panels show task scores on individually zoomed axes; the center shows their unweighted mean.

E.4 Training Variants Produce Distinct Gain Profiles

Figure 10 shows distinct patterns of joint improvement over Raw Trajectories. *MultidocQA* leads on BrowseCompLong, RULER-QA1 (+13.00 pp), and Code Summary (+4.25 pp), and is the only compiled variant to improve Multi-hop Replanning. Gains also extend to Planning Preference and Tool Selection. These co-occurring benefits are consistent with shared demands for integrating contextual information in question answering, code understanding, and planning.

State Tracking instead achieves the largest gain on Bug Localization (+2.86 pp) and ties *MultidocQA* on Tool Selection, while declining on both planning proxies. The two variants thus share benefits for code understanding and tool choice but differ on fault localization and planning. *State Tracking*'s leading gain on RULER-CWE, an aggregation task, also suggests overlap among the targeted long-context capabilities.

Retrieval improves MRCR but regresses on Code Summary and Multi-hop Replanning: retrieval-oriented long-context gains need not accompany stronger agent reasoning. *All* leads on GraphWalks, LongBench V2, and Planning Preference, yet does not preserve the RULER-CWE and Tool Selection gains of the individual variants. The mixture therefore combines strengths on tasks without retaining

[3]https://huggingface.co/datasets/opencompass/mmmlu_lite

every specialized benefit. These findings suggest that trajectory organization shapes overlapping but distinct agent atomic capabilities.

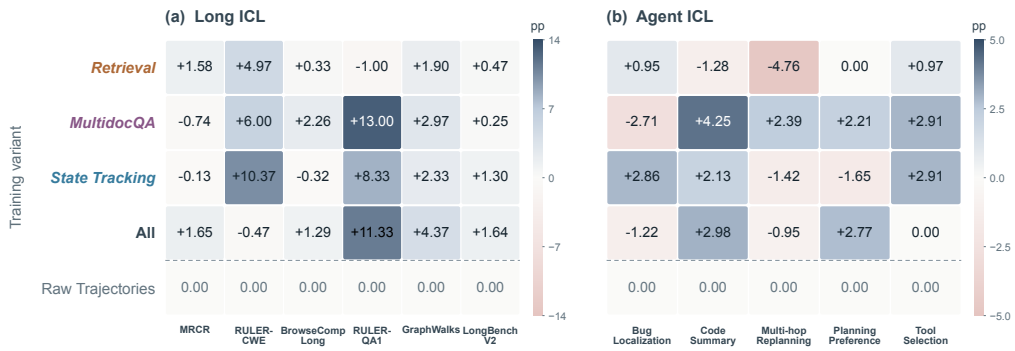


Figure 10: Long-context and Agent ICL accuracy gains over Raw Trajectories (pp). Dashed lines separate the Raw reference row.

E.5 ICL Proxies and Downstream Agent Performance

Figure 11 compares final ICL proxy gains with downstream SFT Agent gains across matched training variants. RULER-Multivalue gains are positively associated with MCP-Atlas gains ($\rho = 0.81$); RULER-QA2 and LongBench V2 provide additional comparisons for multi-document QA and reasoning. Panels (d) and (e) show RULER-VT–SkillsBench and MRCR–SWE-Verified, respectively. The RULER-MultiKey–BrowseComp-ZH comparison is shown separately in Figure 2.

For the additional pairs in Figure 11(d)–(e), SkillsBench and SWE-Verified use BoN@3 scores. RULER-VT averages six lengths (4K–128K); MRCR averages the 2-, 4-, and 8-needle settings at 256K. RULER-MultiKey in Figure 2 averages six lengths (4K–128K).

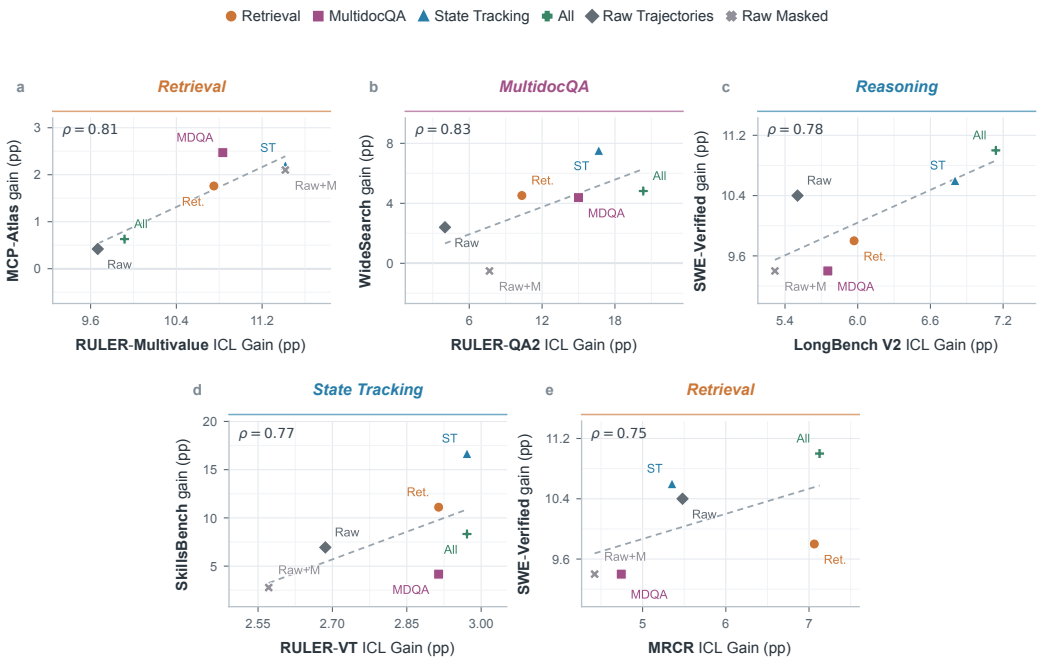


Figure 11: Correlations between long-context ICL gains and downstream agent gains after SFT. Each of the five panels contains six matched training variants. Axes show gains over Qwen3-14B-Base and SFT without mid-training, respectively (pp). Dashed lines are descriptive linear fits; ρ denotes Spearman correlation.

E.6 Post-SFT Long-context–Agent Associations

Different Long-Context Capabilities Align with Different Agent Tasks. Figure 12 shows task-dependent associations after SFT. Retrieval-oriented MRCR correlates positively with BrowseComp and FSC-V2 ($\rho = 0.72$ and 0.56). Evidence aggregation exhibits a different profile: CorpusQA correlates with SWE-Verified and SWE-Pro (0.70 and 0.50), whereas MRCR is negatively correlated with both coding benchmarks. GraphWalks correlates with MCP-Atlas (0.70), but its association with WebArena is weaker (0.30). These complementary associations suggest that different long-context capabilities may be relevant to different downstream task demands. The correspondence is not uniform even within an agent category: MRCR correlates negatively with SealQA (-0.56), and SkillsBench shows no strong positive association with the plotted long-context benchmarks.

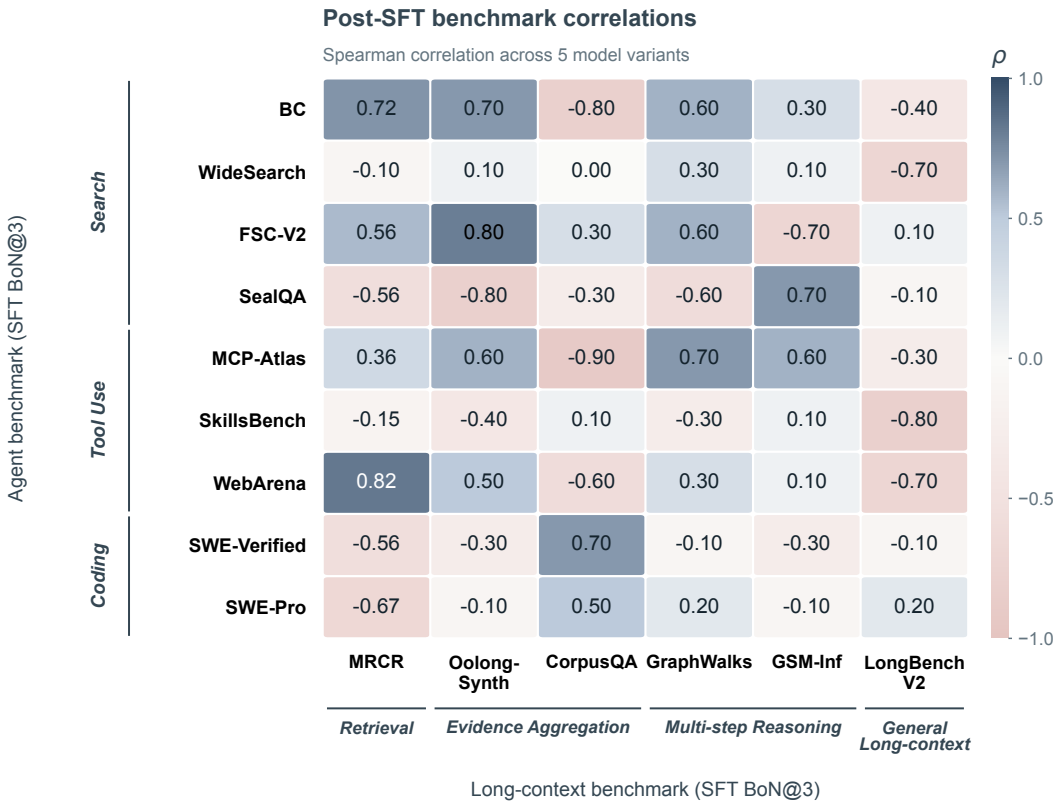


Figure 12: Long-to-Agent correlations after SFT (BoN@3) across five Qwen3-14B variants: Retrieval, MultidocQA, State Tracking, All, and Raw Trajectories. Cells show Spearman ρ with average ranks for ties on a shared $[-1, 1]$ scale.

F TrajLong Data Examples

We present examples of the three TrajLong data types in Tables 17–19. Long contexts are excerpted, with omissions marked as . . . . The Retrieval example retains the search turns needed to answer its three questions, with complete target passages and excerpted solutions. State Tracking retains the key reads and all edits to the queried files; MultidocQA retains selected evidence excerpts. Both use concise questions and shortened or excerpted solutions while preserving the final answers. Identifying details are normalized, and selected prompts and explanations are edited for readability. Non-English text is translated consistently across the context, questions, and answers. Retrieval questions continue the trajectory, whereas State Tracking and MultidocQA questions follow the trajectory or evidence collection.

F.1 Retrieval

This example appends synthetic retrieval turns to an SEO agent trajectory for mid-training. The model locates earlier search results by topic and rank, then reproduces them verbatim with any requested prefix. Thoughts and answers are excerpted for display.

Table 17: A TrajLong (Retrieval) example.

TrajLong / Retrieval	
User	Run today's SEO workflow for example-store.example. Review the saved history and strategy, rotate the target markets across India, Sri Lanka, and Nepal, and avoid duplicate publications.
	... [1 assistant turn omitted; ≈2.9k tokens] ...
Assistant	**Answer:** Today is Monday, so I will check the site's indexing status and recent articles, then prepare the weekly SEO report. **Tool Call:** web_search({ "query": "site:example-store.example", "max_results": 10 })
Observation	[example-store.example (search results)] **Result 1:** finding your goldmine: the ultimate guide to successful dollar store site selection ... **Results 2–3:** ... **Result 4:** 30% off for 3+ items! christmas sale! ... **Result 5:** 30% off for 3+ items! christmas sale! [EMAIL] the ExampleCos collection inspired by the movie **poor things** features a range of stylish clothing and accessories that capture the unique aesthetic of the film. this collection includes vibrant costumes that reflect the whimsical and eclectic fashion seen in the movie, showcasing bold colors and interesting patterns. the clothing is crafted from high-quality materials, ensuring comfort and durability, perfect for cosplay or themed events.
	... [2 assistant turns omitted; ≈2.4k tokens] ...
Assistant	**Tool Call:** web_search({ "query": "site:example-store.example Sri Lanka product mix optimization", "max_results": 5 })
Observation	[Sri Lanka product-mix optimization (search results)] **Result 1:** This paper propose a practical optimization problem of product-mix based on labor capacity, machine capacity, raw materials and demand constrains in a **Sri Lankan tea producing company**. The objective is to profit maximization, satisfying all **Results 2–5:** ...
	... [6 assistant turns omitted; ≈3.6k tokens] ...
Assistant	**Tool Call:** web_search({ "query": "site:example-store.example India inventory turnover", "max_results": 5 })
Observation	[India inventory turnover (search results)] **Result 1:** PDF | On Sep 18, 2021, Aarti Patki published Analysis of Inventory Turnover Ratio and Its Impact on Profitability of Enterprises in Indian Retail Industry Author ... **Results 2–4:** ... **Result 5:** December 30, 2025 · **Analysis of inventory turnover performance within Indian supermarket** Ashish Kumar Bhateja, Abhishek Kumar Shukla Mechanical Engineering, Gulzar Group of Institutions, Khanna ⋯
	... [4 assistant turns omitted; ≈3.4k tokens] ...
(Synthetic retrieval turns for mid-training)	
User	**From the "example-store.example" search, reproduce result 5 (1-indexed) verbatim.**
Assistant	**Thought:** *Results 4 and 5 share the same promotional opening. ... The requested fifth entry is the Poor Things collection passage. Copy it verbatim, without a result label or prefix.* **Answer:** 30% off for 3+ items! christmas sale! [EMAIL] the ExampleCos collection inspired by the movie **poor things** [... omitted for display ...] perfect for cosplay or themed events.

Continued on next page

TrajLong / Retrieval (continued)	
User	**From the "Sri Lanka product-mix optimization" search, reproduce result 1 (1-indexed) verbatim. Prepend 0qM5Sa8J51 followed by a space.**
Assistant	**Thought:** *The first result describes product-mix optimization for a Sri Lankan tea producer. ... Preserve its wording and incomplete ending, and prepend 0qM5Sa8J51 followed by one space.* **Answer:** 0qM5Sa8J51 This paper propose a practical optimization problem of product-mix [... omitted for display ...] in a **Sri Lankan tea producing company**. The objective is to profit maximization, satisfying all
User	**From the "India inventory turnover" search, reproduce result 5 (1-indexed) verbatim. Prepend EVfcnq2nOO followed by a space.**
Assistant	**Thought:** *The fifth result is the supermarket study dated December 30, 2025. ... Include its date and trailing ellipsis, omit the search heading, and prepend EVfcnq2nOO followed by one space.* **Answer:** EVfcnq2nOO December 30, 2025 · **Analysis of inventory turnover performance within Indian supermarket** Ashish Kumar Bhateja, Abhishek Kumar Shukla [... omitted for display ...] Khanna ⋯

F.2 MultidocQA

This example assembles code evidence from an agent trajectory into a collection with evidence IDs and versions. The questions combine evidence across files and versions to identify login endpoints, infer role-dependent navigation, and compare notification-filtering behavior. The original context contains 111 evidence items and approximately 107K tokens; this compact display retains the eight items used by the three questions below.

Table 18: A TrajLong (MultidocQA) example.

TrajLong / MultidocQA

You are given a set of evidence items below, each marked with a unique ID, together with its title and version when available. Read the evidence carefully, identify the items relevant to each question, and answer the question or questions that follow. Evidence IDs are provided to help locate and distinguish evidence items. Use them to keep track of the relevant evidence while solving each question. Follow each question's own output requirements.

... [103 additional evidence items omitted from this compact display] ...

[EVIDENCE id=file-d4ba5ce340e9 | title=AuthRepository.kt | version=1]

```
... class AuthRepository(...) { ...
private val api: AuthApi by lazy { ApiClient.createApi(AuthApi::class.java) }
suspend fun loginWithPassword(...): Result<LoginResponse> = safeCall {
  val response = api.loginWithPassword(LoginRequest(phone, password)); ... }
suspend fun loginWithSms(...): Result<LoginResponse> = safeCall {
  val response = api.loginWithSms(SmsLoginRequest(phone, code)); ... }
... }
```

[EVIDENCE id=file-ce5214a29b14 | title=AuthApi.kt | version=1]

```
... interface AuthApi {
@POST("/api/v1/auth/tenant/login") suspend fun loginWithPassword(...) ...
@POST("/api/v1/auth/sms/send") suspend fun sendSmsCode(...) ...
@POST("/api/v1/auth/sms/login") suspend fun loginWithSms(...) ...
... }
```

[EVIDENCE id=file-f879713c7014 | title=AppNavGraph.kt | version=1]

```
... LoginScreen(onLoginSuccess = {
val user = authState.user
val homeRoute = when { user?.isAdmin() == true -> Screen.AdminHome.route
  else -> Screen.WorkerHome.route }
navController.navigate(homeRoute) { popUpTo(Screen.Login.route) { inclusive = true } }
}) ...
```

Continued on next page

TRAJLONG / MultidocQA (continued)

[EVIDENCE id=file-8c0dc44e8e97 | title=NotificationApi.kt | version=1]
... @GET("/api/v1/notifications") suspend fun getNotifications(
@Query("page") page: Int = 1, @Query("limit") limit: Int = 20,
@Query("is_read") isRead: Boolean? = null): PaginatedResponse<NotificationResponse> ...

[EVIDENCE id=file-043ad1aa30d9 | title=NotificationApi.kt | version=2]
... @GET("/api/v1/notifications") suspend fun getNotifications(
@retrofit2.http.Query("page") page: Int = 1, @retrofit2.http.Query("limit") limit: Int = 20
): ApiResponse<List<Notification>> ...

[EVIDENCE id=file-302481661e72 | title=NotificationApi.kt | version=3]
... @GET("/api/v1/notifications") suspend fun getNotifications(
@Query("page") page: Int = 1, @Query("limit") limit: Int = 20,
@Query("is_read") isRead: String? = null): PaginatedResponse<Notification> ...

[EVIDENCE id=file-fd9d2fee6a0c | title=notificationController.js | version=1]
... async function listNotifications(req, res) {
const { type, is_read, page = 1, limit = 20 } = req.query
const { user_id, user_type } = req.user; const where = { user_id, user_type }
if (type) where.type = type
if (is_read !== undefined && is_read !== '') {
  where.is_read = is_read === '1' || is_read === 'true'
}
... Notification.findAndCountAll({ where, ... }) ... }

[EVIDENCE id=file-4150517855d6 | title=User.kt | version=1]
... data class User(... val role: String, ...) {
fun isAdmin(): Boolean = role == "admin"; ... }

Question 1. Which POST endpoints does AuthRepository use for password and SMS login? Return the two paths as a JSON array in alphabetical order.
Reason: *The two repository methods must first be mapped to the corresponding Retrofit methods. In file-d4ba5ce340e9, loginWithPassword calls api.loginWithPassword(LoginRequest(phone, password)), while loginWithSms calls api.loginWithSms(SmsLoginRequest(phone, code)). ... In file-ce5214a29b14, the @POST annotation on AuthApi.loginWithPassword specifies "/api/v1/auth/tenant/login". The annotation on AuthApi.loginWithSms specifies "/api/v1/auth/sms/login". The same interface also declares "/api/v1/auth/sms/send", but that endpoint belongs to sendSmsCode and* ***only sends a verification code****; it is not the SMS login endpoint requested by the question. ... Their common prefix is "/api/v1/auth/", so alphabetical ordering places "sms" before "tenant". ...*
Answer: ["/api/v1/auth/sms/login", "/api/v1/auth/tenant/login"]

Question 2. After a user with role 'designer' logs in through LoginScreen, which route is selected? Give the popUpTo target and its inclusive setting.
Reason: *The role check and the navigation callback are defined in separate evidence items. In file-4150517855d6, User.isAdmin() returns role == "admin". A user whose role is "designer" therefore does not satisfy the administrator check. ... In file-f879713c7014, the LoginScreen onLoginSuccess callback reads authState.user and chooses homeRoute with a when expression. It selects Screen.AdminHome.route only when user?.isAdmin() == true; otherwise, it selects Screen.WorkerHome.route.* ***The designer user follows this second branch.*** *... The callback then calls navController.navigate(homeRoute) with popUpTo(Screen.Login.route) { inclusive = true }. Thus the navigation pops the back stack up to and including the login destination, removing the login screen. ...*
Answer: Screen.WorkerHome.route; popUpTo(Screen.Login.route), inclusive = true.

Continued on next page

TRAJLONG / MultidocQA (continued)

Question 3. For GET /api/v1/notifications, which statements about the Android API versions and backend is_read handling are correct? Select all that apply; return comma-separated letters in alphabetical order.
(A) Android v2 declares isRead: Int? = 0 as a query parameter.
(B) Android v1 declares isRead: Boolean? = null as a query parameter.
(C) Android v3 declares isRead: String? = null as a query parameter.
(D) Omitting is_read makes the backend return only unread notifications.
(E) Android v2 omits the is_read query parameter from getNotifications.
(F) The backend applies an is_read filter only when the parameter is neither undefined nor empty, interpreting the strings '1' and 'true' as true.
Reason: *The three Android evidence items must be checked separately because they contain different versions of the same interface. In version 1 (file-8c0dc44e8e97), getNotifications declares @Query("is_read") isRead: Boolean? = null, making (B) correct. Version 2 (file-043ad1aa30d9) declares only page and limit in that method, so (E) is correct and (A) is false. ... Version 3 (file-302481661e72) declares @Query("is_read") isRead: String? = null, making (C) correct. The backend logic is in listNotifications in file-fd9d2fee6a0c. ... It adds where.is_read only when is_read !== undefined && is_read !== ''. Inside that condition, it evaluates is_read === '1' || is_read === 'true'. This directly supports (F). When the request omits is_read, the condition is false and listNotifications adds* ***no read-status filter****. Omission therefore does not mean "unread only", so (D) is false. ...*
Answer: B,C,E,F

F.3 State Tracking

This example follows a coding agent adding font-size controls. The questions trace file states across reads and edits, distinguishing initial content from later changes and identifying the last operation that wrote a target fragment. All successful Edit and Write tool calls on the queried files are retained; omitted turns concern other files or actions unrelated to the answers. Tool-call arguments marked as abridged are shortened only for display.

Table 19: A TrajLong (State Tracking) example.

TrajLong / State Tracking	
The following is a historical agent trajectory enclosed between [BEGIN_OF_TRAJECTORY] and [END_OF_TRAJECTORY].	
[BEGIN_OF_TRAJECTORY]	
System	You are an AI coding assistant.
	... [20 assistant turns omitted; ≈28.0k tokens] ...
User	Can we adjust the font size? It looks a little too small.
Assistant	**Tool Call:** Read({"file_path": ".../src/renderer/index.css"})
Observation	... 10→ :root { 11→ /* Cyberpunk Color Tokens */ 12→ --background: #0a0a0f; 13→ --foreground: #e0e0e0; ...
	... [3 assistant turns omitted; ≈9.8k tokens] ...
Assistant	**Answer:** Add separate UI and terminal font-size variables. **Tool Call (abridged):** Edit({"file_path": ".../src/renderer/index.css", "old_string": ":root { /* Cyberpunk Color Tokens */", "new_string": ":root { /* Font Size Tokens (adjustable via UI) */ --ui-font-size: 12px; --terminal-font-size: 12px; /* Cyberpunk Color Tokens */"})
Observation	Updated index.css successfully.
Assistant	**Answer:** Apply the UI font-size variable to the body rule. **Tool Call (abridged):** Edit({"file_path": ".../src/renderer/index.css", "old_string": "body { font-family: ...; ... }", "new_string": "body { font-family: ...; font-size: var(--ui-font-size); ... }"})
Observation	Updated index.css successfully.
	... [3 assistant turns omitted; ≈1.4k tokens] ...
Assistant	**Answer:** Inspect TerminalGrid before updating terminal font sizes. **Tool Call:** Read({"file_path": ".../src/renderer/components/TerminalGrid.jsx"})
Observation	... 126→ setTerminalOrder((prev) => { 127→ let rows = prev.map((row) => row.filter((id) => id !== agentId)); 128→ rows[targetRowIdx] = [129→ ...rows[targetRowIdx].slice(0, targetColIdx), 130→ agentId, 131→ ...rows[targetRowIdx].slice(targetColIdx), 132→]; 133→ return rows.filter((r) => r.length > 0); 134→ }); ...
	... [4 assistant turns omitted; ≈2.2k tokens] ...
Assistant	**Answer:** Expose updateAllFontSize to update existing terminals. **Tool Call (abridged):** Edit({"file_path": ".../src/renderer/components/TerminalGrid.jsx", "old_string": "useImperativeHandle(ref, () => ({ ... }));", "new_string": "useImperativeHandle(ref, () => ({ ... updateAllFontSize: (size) => { Object.values(termRefs.current).forEach((t) => t?.updateFontSize?.(size)); }, }));"})
Observation	Updated TerminalGrid.jsx successfully.
	... [33 assistant turns omitted; ≈27.8k tokens] ...
[END_OF_TRAJECTORY]	

Continued on next page

TRAJLONG / State Tracking (continued)

Return only {"operation":"...","successful_index_in_file":N}. Count only successful Edit, Write, and Delete tool calls, separately by operation type and file, starting at 1. Read and Bash tool calls (including git commands), failed calls, and operations on other files do not count. Use **anchor** with index 0 for a fragment present at the trajectory's start.

Question 1: Identify the first appearance of this fragment in **TerminalGrid.jsx**. Return **anchor** if it was present at the start; otherwise, identify the successful edit or write that first introduced it.
[TARGET_FRAGMENT]
let rows = prev.map((row) => row.filter((id) => id !== agentId));
rows[targetRowIdx] = [
[END_OF_TARGET_FRAGMENT]
Reason: *The first read of TerminalGrid.jsx already shows the target at lines 127–128, before any successful edit or write to this file. Its only later edit adds updateAllFontSize to useImperativeHandle and leaves the target fragment unchanged. The read observes existing content rather than creating it. Therefore, the fragment belongs to the initial state: anchor, index 0.*
Answer: {"operation":"anchor","successful_index_in_file":0}

Question 2: This fragment remains in **index.css** at the end of the trajectory. Which successful edit or write last wrote it?
[TARGET_FRAGMENT]
/* Font Size Tokens (adjustable via UI) */
--ui-font-size: 12px;
--terminal-font-size: 12px;
[END_OF_TARGET_FRAGMENT]
Reason: *The initial read of index.css has no font-size token block. Successful edit 1 inserts the comment and both 12px variables under :root. Successful edit 2 only adds font-size: var(--ui-font-size); to the body rule, leaving those tokens intact. No later edit or write rewrites the target fragment. Thus, edit 1 is the last operation that wrote this fragment, even though edit 2 is the file's last edit.*
Answer: {"operation":"edit","successful_index_in_file":1}