11-768 · AI AGENTS | TRAINING 1

Supervised Fine-Tuning for Agents

Turning recorded trajectories into weights

Carnegie Mellon University
Language Technologies Institute

Yueqi Song
Language Technologies Institute

Today

- Weight updates as a capability
- Trajectories as token sequences
- Choosing the trajectories
- One format for many datasets
- Running the training
- Knowing it worked
- Handing off to RL

A trajectory as recorded data

- The agent received an issue and a couple of tools in a sandbox: **bash** and a file editor.
- It produced thoughts, commands, and the observations those commands returned.
- The repository's own tests decided whether the run succeeded.

Pan et al., 2024 · SWE-Gym, § 4.2 ↗

Methods of updating the agent

LOCATION	PROS	CONS
Context window SEP 1	high fidelity to what happened	costly, noisy, does not decide what matters
External artifacts SEP 3	inspectable, editable, retrievable, portable	must be induced, selected, and maintained
Model weights THIS LECTURE	faster inference, broad behavior change	slower updates, opaque, model-specific

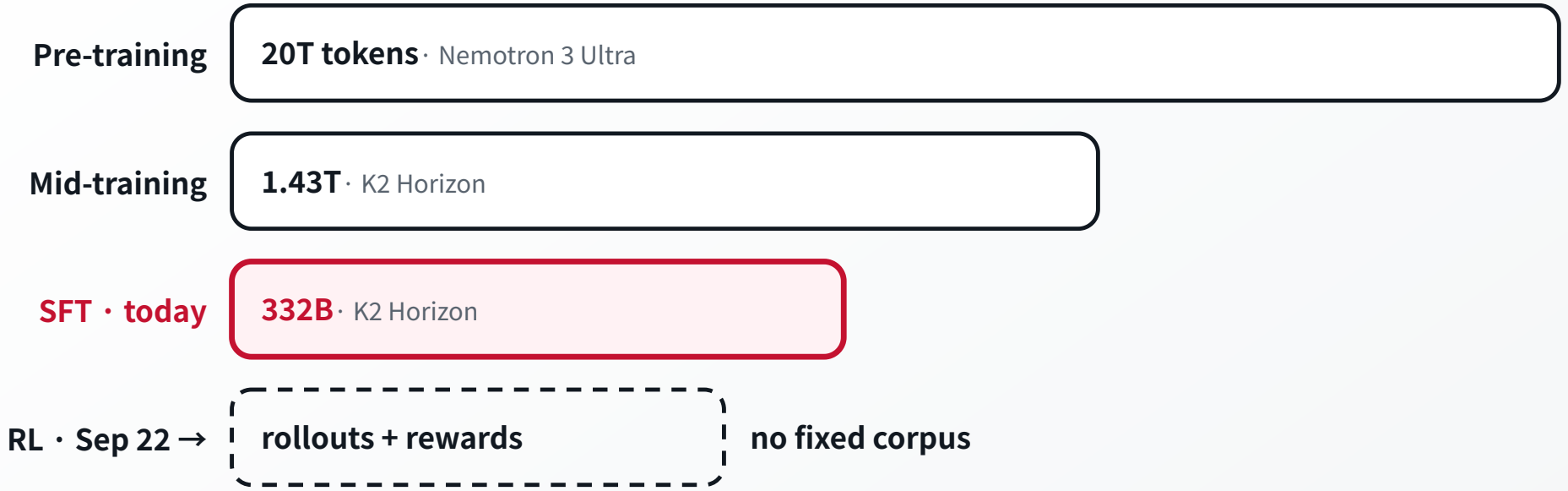
- Change the weights once and every call after that gets the new behavior, with nothing extra in the prompt.
- The first two rows update in seconds. This row costs a training run every time you change it.
- And you cannot open the weights to see what changed, or move the change to a different model.

What the trajectories teach

- How to write a tool call this harness can actually run.
- How to keep going when the conversation gets long, because the trajectories it learned from were long.
- What a tool result looks like, so it waits for one instead of writing one itself.
- And whatever else was in the data: extra commentary, repeated commands, one way of solving a problem where several would work.

The full training pipeline

Drawn to token scale: every step down is fewer tokens, each one more heavily curated.



Nemotron 3 Ultra ↗ K2 Horizon · Training Overview ↗

Cold start before reinforcement learning

"Cold start" is the checkpoint RL begins from. Today's stage is how K3 makes that checkpoint good.

- In Kimi K3, the model learns to call tools and finish long tasks during SFT, before any RL.
- Some might ask: why not just RL, without SFT? DeepSeek-R1-Zero ran RL directly on the base model, with no SFT at all.
- Their very next model put SFT back in: R1 starts from a few thousand curated examples, because R1-Zero's answers mixed languages and were hard to read.

"The SFT stage establishes a high-quality cold-start policy for the subsequent RL stage."

Kimi Team, 2026 · Kimi K3, § 4.1 ↗ DeepSeek-R1 ↗

SFT and RL, side by side

	SUPERVISED FINE-TUNING	REINFORCEMENT LEARNING
learns from	recorded trajectories, whoever produced them	its own rollouts, scored by a reward
signal	a target for every token	one number per rollout
needs	data and GPUs	live environments, sandboxes, a checkable reward
good at	learning tool calls, formats, long tasks	sharpening what the model can already attempt
stuck when	the demonstrations are wrong, narrow, or used up	the start is too weak, or the reward cannot be checked

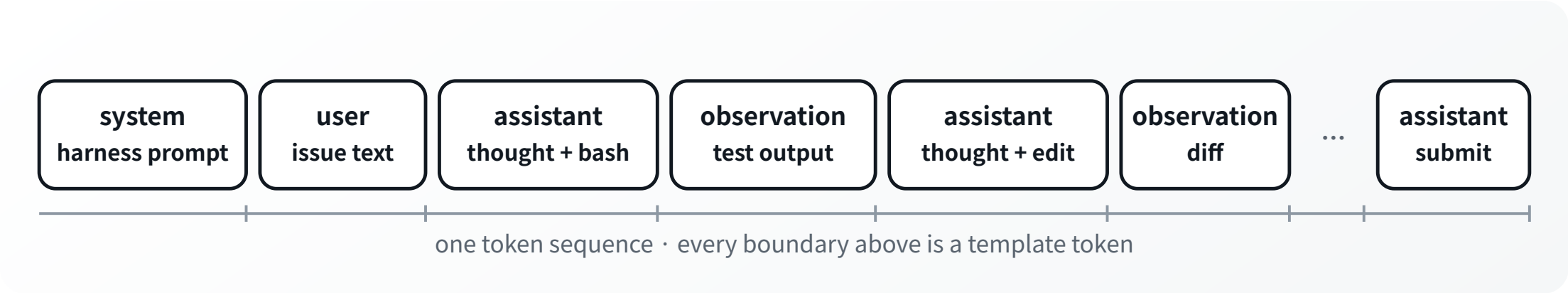
OpenThoughts-Agent, Table 11 ↗ Kimi K3 ↗ Kimi K2, § 3.2 ↗

The training lectures

TOPIC	WHEN
Supervised fine-tuning	today
Reinforcement learning	Sep 22 · Sep 29 · Oct 1

Trajectories as token sequences

The chat template applied to the trajectory




```
<|im_start|>tool ↵ 3·failed,·41·passed ↵ <|im_end|> ↵ <|im_start|>assistant
↵ The·test·compares·floats.·Fix·it. ↵ <tool_call> ↵ str_replace(...) ↵
</tool_call> ↵ <|im_end|>
```

"·" = space, "↵" = newline

- Feed the trajectory through the template and you get one string of tokens.
- The `<|...|>` markers are single special tokens. The rest is what the agent wrote and saw.

Transformers · chat templating ↗ Kimi K3 · XTML, § 4.1.1 ↗

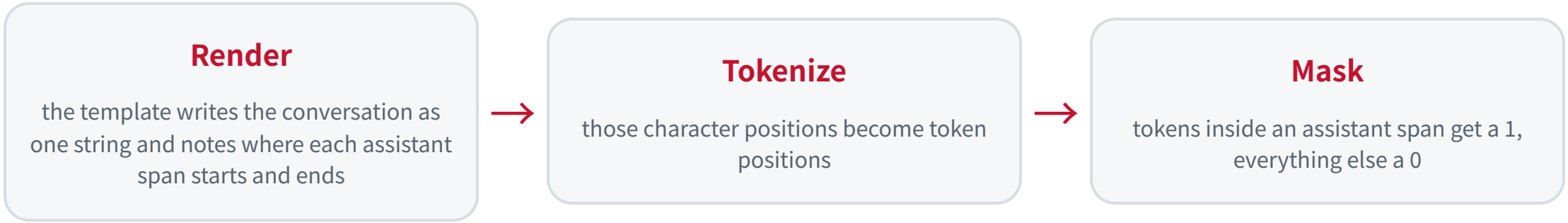
Which tokens carry the loss

Cross-entropy over the assistant tokens only

$$\mathcal{L}(\theta) = -\sum_{t\,:\,m_t=1} \log p_\theta(x_t \mid x_{<t})$$

- Mask: one bit per token, saying whether that token's prediction counts.
- Minimizing it raises the probability of each recorded action, given everything before it.
- Observations are conditioning context, not prediction targets.

How the mask is built



- The template does the bookkeeping as it writes: where assistant text starts, where it stops.
- By the time training starts, the roles are gone. The trainer sees the string and the bits, nothing else.

Transformers · apply_chat_template ↗

Masking controls across frameworks

FRAMEWORK	WHAT YOU SET	HOW THE MASK IS PRODUCED
TRL	`assistant_only_loss=True`	the template marks the assistant spans as it renders
Axolotl	`roles_to_train`, `train_on_eos`	finds each turn's role header in the rendered string
LLaMA-Factory	`train_on_prompt`, `mask_history`	tags whole turns as train-or-ignore while encoding

- Same decision in all three. Only TRL puts it in the template.

TRL · SFTTrainer ↗ Axolotl · conversation formats ↗ LLaMA-Factory · data args ↗

The end-of-turn token

<|im_start|>assistant ↵ I'll·run·the·failing·test. ↵ <tool_call> ↵
bash(cmd="pytest·-q") ↵ </tool_call> ↵ <|im_end|> ↵

"·" = space, "↵" = newline · red = loss computed, gray = context only

The role header is supplied at inference. It is never a target.

The model produces the stop token, so it has to be inside the mask.

"End of turn" is three events, not one: end this message, hand off to a tool, or finish the task. Kimi K3's template spells them apart with separate tokens:

[open]message·role=assistant[sep] [open]think[sep] … [close]think[sep]
[open]response[sep] Use·math.isclose. [close]response[sep] [end_of_msg]

"the model may not learn to stop."

TRL · training chat templates ↗ TRL · SFTTrainer source ↗ Kimi K3 · XTML, § F ↗

Masking the reasoning block

<|im_start|>assistant ↵ <think> ↵ The·assert·compares·floats. ↵ </think> ↵
Use·math.isclose. ↵ <|im_end|>

"·" = space, "↵" = newline · red = loss computed, gray = context only

- Two switches, set separately: does the reasoning stay in the context, and does it get trained on.
- In the context, the reasoning explains the action that follows. Trained on, it becomes the style the model itself writes.
- Nemotron keeps budget-truncated reasoning in the context, and masks the artificial cut-off out of the loss.

NVIDIA, 2026 · Nemotron 3 Ultra, § 3.1.1 ↗

Weighting inside the assistant turn

Models	Modules		Single Turn	Multi Turn
	SSB	HDR		
Qwen2.5-Coder-Inst+SFT	×	×	81.73	38.25
	✓	×	82.20	41.62
	×	✓	83.85	43.12
	✓	✓	84.00	47.00
Llama-3.2-3B-Inst+SFT	×	×	73.67	33.25
	✓	×	76.82	34.62
	×	✓	78.28	35.75
	✓	✓	78.99	36.12

Table 5: Performance comparison of Qwen2.5-Coder-Inst and Llama-3.2-3B-Inst, across different components of modules (SSB and HDR) in Single-Turn and Multi-Turn on BFCL.

Two components from one paper: SSB is a learned weight that rebalances reasoning tokens against tool-call tokens in the loss, and HDR resamples the hard examples.

- Long reasoning dominates the loss, and the short tool call is what you actually need correct.
- Rebalancing helps more on multi-turn than on single-turn, on both base models.

Hao et al., 2025 · BalanceSFT, Table 5 ↗

Samples versus tokens

Capability	Sample weight	Token weight
STEM and Coding	56%	89%
Agentic Capability	11%	9%
General Helpfulness and Safety	33%	2%

Table 10. Consolidation SFT data mixture composition by capability.

They chose the mixture by counting examples. The loss is computed per token.

- The mixture was balanced by counting examples, but STEM and coding traces run far longer, so they take nearly all the tokens.
- Whatever you balance by counting trajectories, check what it turned into in tokens.

Microsoft AI, 2026 · MAI-Thinking-1, Table 10 ↗

Open problems in supervising a trajectory

- We could not find a published experiment that changes the mask for agent SFT and measures what happens.
- The careful masking experiments are single-turn instruction tuning, not agents.
- The one case where unmasking the prompt helped had long prompts, short answers, and little data. Agent trajectories are the opposite on all three.
- At 8B scale this experiment is cheap, and it is sitting there unrun.

Chatterjee et al., 2025 · WIT ↗ Shi et al., 2024 · Instruction Modelling ↗

Choosing the trajectories

Where the trajectories come from

WHO RAN THE TASK	YOU KEEP EVERYTHING	YOU KEEP THE RUNS THAT PASSED
a stronger model ran it	plain distillation	SWE-Gym · Nemotron · OT-Agent
the model ran it itself	MAI, on its own reasoning runs	expert iteration · Sep 22

- Most agent SFT data sits in the top right box: a stronger model writes better trajectories than the student could, and the task's own tests make keeping the good ones cheap.
- In the bottom row the model makes its own training data. Repeat that loop and you get expert iteration, which will be covered in future lectures.

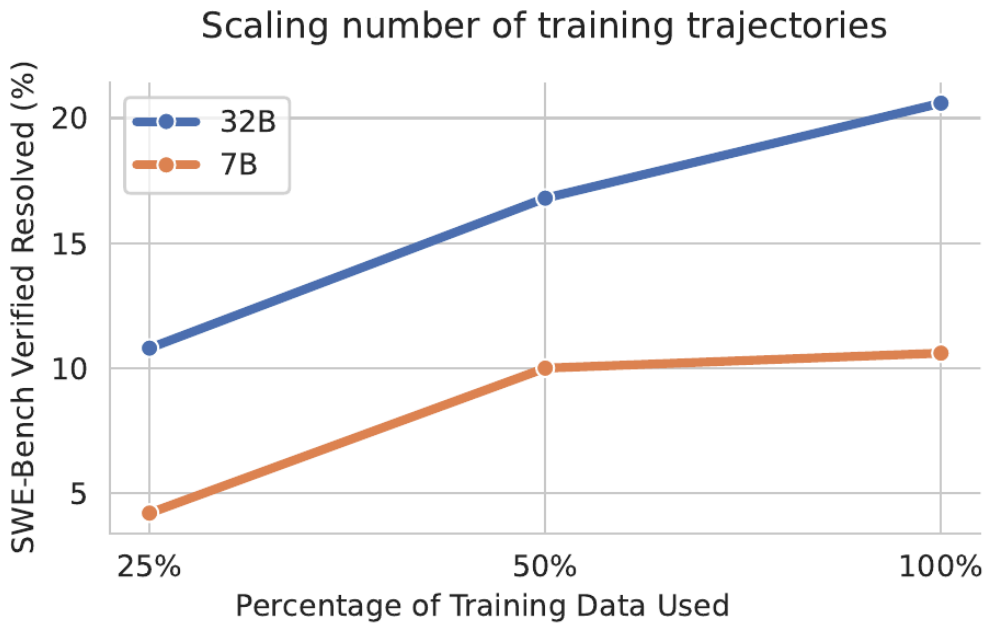
Distillation from a stronger model

Model Size	Empty Patch (%, ↓) zero-shot	fine-tuned	Δ	Stuck in Loop (%, ↓) zero-shot	fine-tuned	Δ	Avg. Turn(s) zero-shot	fine-tuned	Δ	Resolve Rate (%, ↑) zero-shot	fine-tuned	Δ
SWE-Bench Lite (300 instances)												
7B	40.3	29.7	-10.7	47.0	31.0	-16.0	20.3	22.2	+1.9	1.0 (± 1.0)	10.0 (± 2.4)	+9.0
14B	49.7	**18.1**	-31.6	31.7	27.1	-4.6	23.2	21.4	-1.8	2.7 (± 1.9)	12.7 (± 2.3)	+10.0
32B	**27.0**	**18.1**	-8.9	**16.7**	**18.1**	+1.5	15.5	29.3	+13.9	**3.0** (± 1.4)	**15.3** (± 2.5)	**+12.3**
SWE-Bench Verified (500 instances)												
7B	45.8	33.8	-12.0	39.6	**21.0**	-18.6	21.9	35.3	+13.4	1.8 (± 1.1)	10.6 (± 2.1)	+8.8
14B	44.9	14.5	-30.4	32.1	21.3	-10.7	25.5	30.1	+4.6	4.0 (± 1.6)	16.4 (± 2.0)	+12.4
32B	**9.5**	**13.8**	+4.3	**29.4**	23.8	-5.6	24.6	31.6	+7.0	**7.0** (± 1.3)	**20.6** (± 2.1)	**+13.6**

- SEP 10 GPT-4o and Claude ran the SWE-Gym tasks; 491 runs passed the repository tests.
- A 32B Qwen model was fine-tuned on those 491 runs alone.
- Its scores on both SWE-Bench splits rose by more than ten points.

Pan et al., 2024 · SWE-Gym, § 4.2 and Table 3 ↗

Trajectory count and saturation

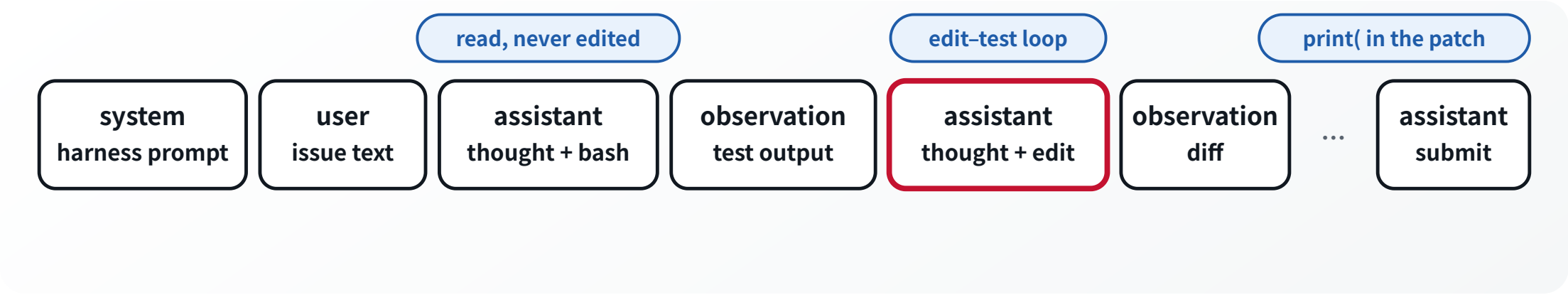


Same task set throughout. Only the number of sampled trajectories changes.

- Accuracy was still rising when the sampling budget ran out.
- They had tasks to spare. What they ran out of was compute to sample more runs.
- Every point in this plot uses the same tasks. What happens when you add new tasks comes a few slides later.

Pan et al., 2024 · SWE-Gym, § 5.2 and Figure 5 ↗

Undesirable behaviors in successful rollouts



- Every run on this slide passed its tests.
- But look at how: one edited and re-ran tests in a loop for most of its turns, another left print statements in the final patch.
- The loss copies every action, good or not. Train on these runs and the model learns the mess together with the fix.

"would teach undesirable behaviors if used directly as SFT data"

NVIDIA, 2026 · Nemotron 3 Ultra, § 3.1.1 ↗

Filtering rollouts by shape

Rank	Trajectory Filter	Benchmarks			Average	
		SWE-bench Verified (100)	OT-TBLite	Terminal-Bench 2.0	Raw	Normalized
1	Min turns ≥ 5	**29.00** 1.56	**19.10** 1.57	**11.61** 1.30	**19.90**	**+1.25**
2	Filter timeouts	26.67 1.70	**18.03** 1.60	**10.49** 1.18	18.39	-0.35
3	Filter subagent traces	23.00 1.63	**17.31** 1.58	**10.86** 1.35	17.06	-0.90

Table 7: **Filtering agent rollouts: keeping longer trajectories helps.** The minimum-turns filter outperforms timeout and subagent filters across all three benchmarks.

- The best filter here just counts turns and drops every run shorter than five.
- Turn count says how much work a run took, not whether the work was any good.
- A four-turn clean fix gets dropped. A forty-turn mess gets kept. On average it still helped.

Raoof et al., 2026 · OpenThoughts-Agent, Table 7 ↗

Choosing the teacher

Rank	Teacher Model	Benchmarks			Average	
		SWE-bench Verified (100)	OT-TBLite	Terminal-Bench 2.0	Raw	Normalized
1	GLM 4.7 (Quantized)	28.00 1.76	**17.86** 1.60	**8.61** 1.40	**18.16**	**+0.73**
2	Kimi K2.5	**33.33** 1.83	14.19 1.51	**8.24** 1.06	**18.59**	**+0.66**
3	GLM 5	**33.00** 1.60	14.30 1.78	**7.50** 1.18	**18.27**	**+0.66**
4	GLM 4.6 (Quantized)	26.00 1.67	**16.99** 1.80	6.37 1.24	16.45	+0.08
5	GPT-5.3-Codex	21.67 1.33	10.42 1.55	3.75 0.99	11.94	-1.47

Table 6: **Teacher model ablation: stronger model $\neq$ better teacher.** Despite GPT-5.3-Codex being the strongest model on these benchmarks, it is the weakest teacher, underperforming GLM 4.7 AWQ by $\sim$5pp on Terminal-Bench 2.0.

- The paper reports GPT-5.3-Codex was the strongest of these five models on the benchmarks themselves. Its runs produced the weakest student.
- Leaderboard rank told them the wrong thing here. Trying two teachers and keeping the better student is cheap by comparison.

Raoof et al., 2026 · OpenThoughts-Agent, Table 6 ↗

Task sources

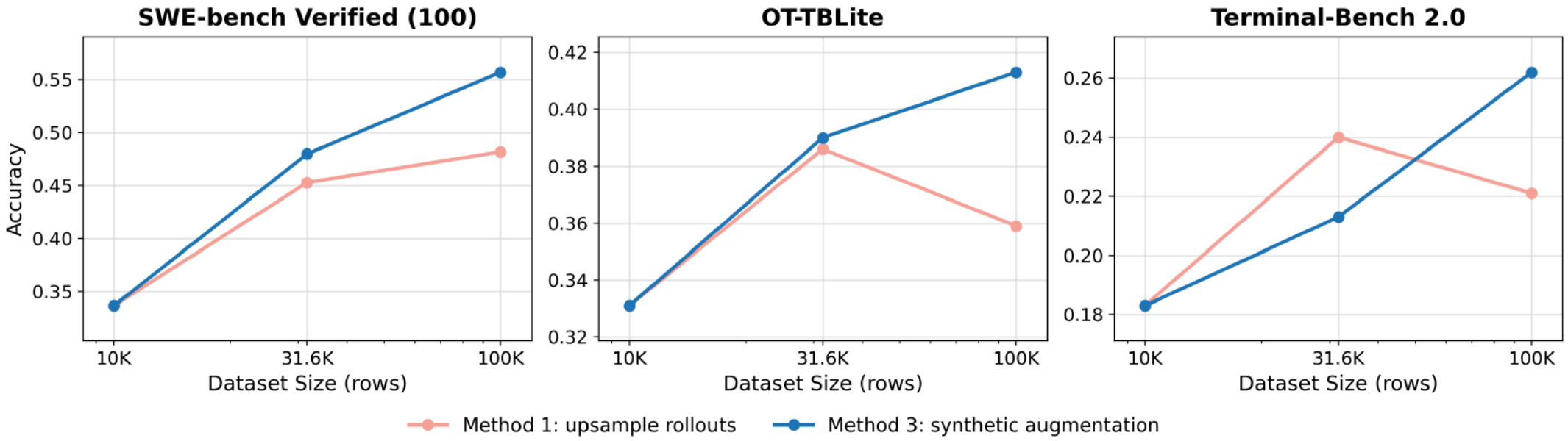
		Benchmarks			Average	
Rank	Strategy	SWE-bench Verified (100)	OT-TBLite	Terminal-Bench 2.0	Raw	Normalized
1	SWESmith	**32.33** 1.83	17.63 1.63	6.37 1.18	**18.78**	**+1.92**
2	StackExchange SuperUser	13.33 1.41	16.68 1.63	**10.86** 1.35	13.62	+1.51
3	StackExchange Tezos	16.33 1.41	16.94 1.83	9.36 1.06	14.21	+1.45
4	IssueTasks	24.00 1.76	16.44 1.49	6.74 1.24	15.73	+1.37
		...				
95	AgentTuning-OS	0.00 0.00	5.64 1.12	0.37 0.37	2.00	-2.31

Table 2: **Task source choice has the largest spread of any pipeline stage.** Issue-resolution tasks and human-written infrastructure questions dominate the top, but improvements are spiky across benchmarks. Full ranking in Appendix A.1.

- The choice of task source changed results the most.
- Different sources help different benchmarks, so spend your effort here before tuning anything else.
- Mix the best four to eight sources. Going wider than that did not help them.

Raoof et al., 2026 · OpenThoughts-Agent, Table 2 ↗

Scaling the data



- Both curves start from the same 10K dataset and differ only in where the extra data came from.
- The pink curve adds more runs of the same tasks, and it flattens out. The blue curve adds new tasks, and it keeps climbing.

Raoof et al., 2026 · OpenThoughts-Agent, Figure 3 ↗

Discussion

Three runs of the same task just came back. You are deciding what goes into the training set.

> **A.** A short run that solved the issue in four turns.

> **B.** A long run that made a wrong edit, saw the test fail, and recovered.

> **C.** A run that solved the issue after eleven attempts that changed nothing.

- Which of these do you keep?
- For the ones you keep, which actions carry the loss?
- Which of your answers would a minimum-turns filter get wrong?

One format for many datasets

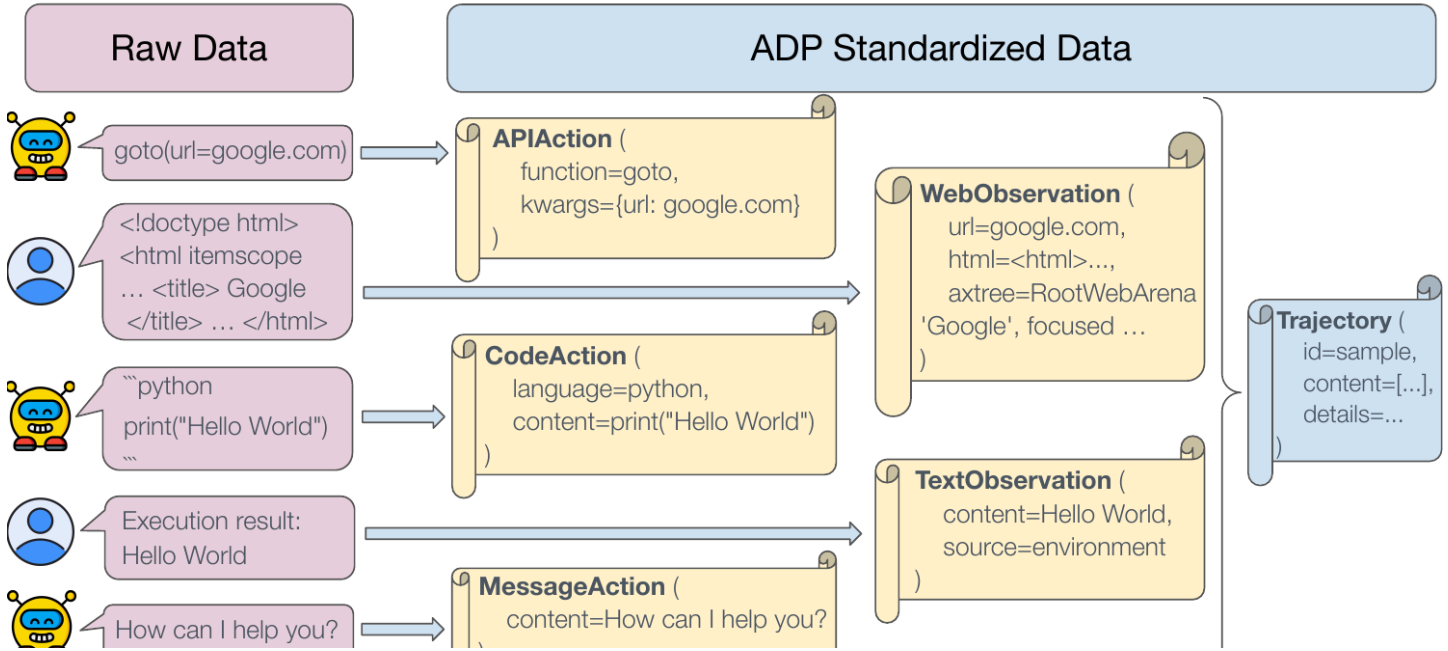
Heterogeneity across agent datasets

Dataset	Variety	Count	Source	Note
AgentInstruct (Zeng et al., 2023)	C T W	1.9K	synthetic	Mixture of Browsing, Database, OS, etc.
Code-Feedback (Zheng et al., 2024a)	C	66.4K	manual	Code generation with runtime feedback loops
CodeActInstruct (Wang et al., 2024b)	C	7.1K	synthetic	Code generation and tool use with execution
Go-Browse (Gandhi & Neubig, 2025)	W	9.5K	rollout	Structured exploration web rollouts
Mind2Web (Deng et al., 2023)	W	1.0K	manual	Human web demos on real websites
Nebius SWE Trajectories (Golubev et al., 2024)	S	13.4K	rollout	SWE-agent trajectories from Nebius relying solely on open-weight models
NNetNav-live (Murty et al., 2024)	W	5.0K	rollout	Retroactively labeled live web exploration
NNetNav-wa (Murty et al., 2024)	W	4.2K	rollout	Retroactively labeled WebArena exploration
openhands-feedback (All Hands AI, 2024)	C T W	0.2K	rollout	Recorded OpenHands agent trajectories with human feedback
Orca Agentinstruct (Mitra et al., 2024)	T	1046.1K	synthetic	Large-scale synthetic tool-use instructions data
SWE-Gym (Pan et al., 2025)	S	0.5K	rollout	Agent trajectories solving real GitHub repo tasks
SWE-smith (Yang et al., 2025b)	S	5.0K	synthetic	Trajectories of agents on synthesized bug-fix tasks
Synatra (Ou et al., 2024)	W	99.9K	synthetic	Synthetically created web demos of tutorials

- Each dataset was built by a different group, in its own format.
- For a web page, some save the HTML and others save the accessibility tree.
- To train on two of them together, you first write two customized converters.

Song et al., 2026 · Agent Data Protocol, Table 1 ↗

One representation for a trajectory



- You read each dataset once and rewrite it as typed actions and observations.
- For the web page we just saw, keep the HTML and the accessibility tree, instead of picking one.
- Each dataset converts once, each harness once, not once per pair.

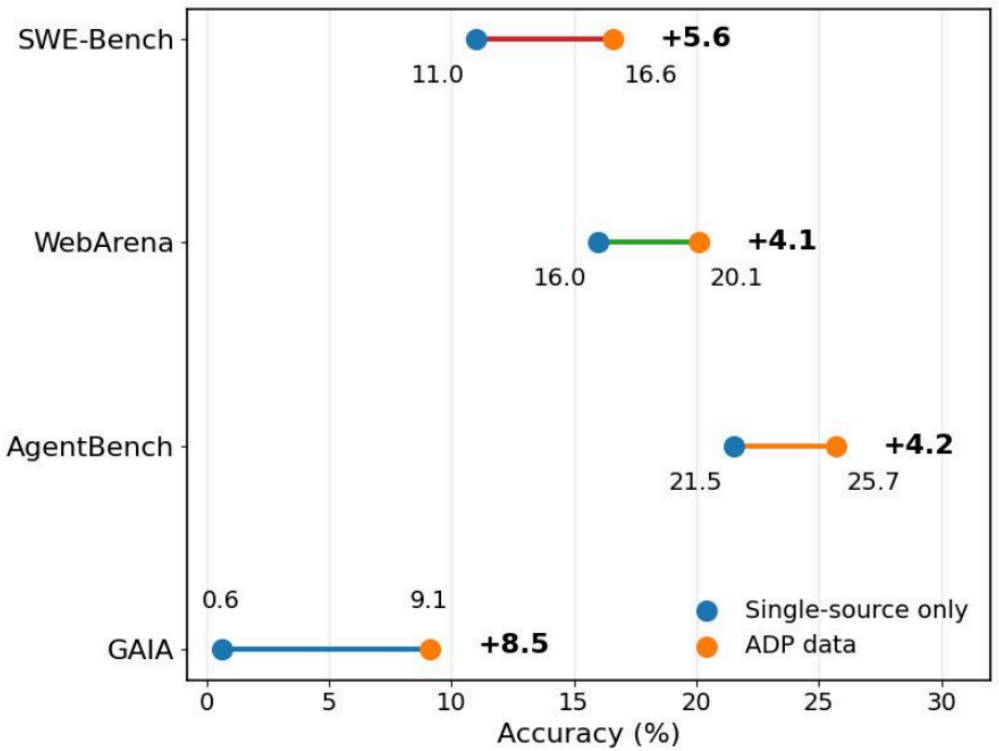
Song et al., 2026 · Agent Data Protocol, ICLR oral ↗

Harness-specific rendering

- Once the data is in one format, you still have to write it back out for the harness you are training for.
- OpenHands, SWE-Agent and AgentLab take different actions, so the same trajectory comes out looking different in each.
- This is also where you set the system prompt and decide what to do when the context gets long.
- Then you check the result: do the tool calls parse, does each call come with a thought, does the conversation end the way it should.

Song et al., 2026 · Agent Data Protocol, § 3.3 ↗

Diversity versus task-specific data



- Same harness, same model, same evaluation. Only the training mixture changes.
- Train on the mixture and you beat the matching single-domain set, even on that domain's own benchmark.

Song et al., 2026 · Agent Data Protocol, cross-task transfer ↗

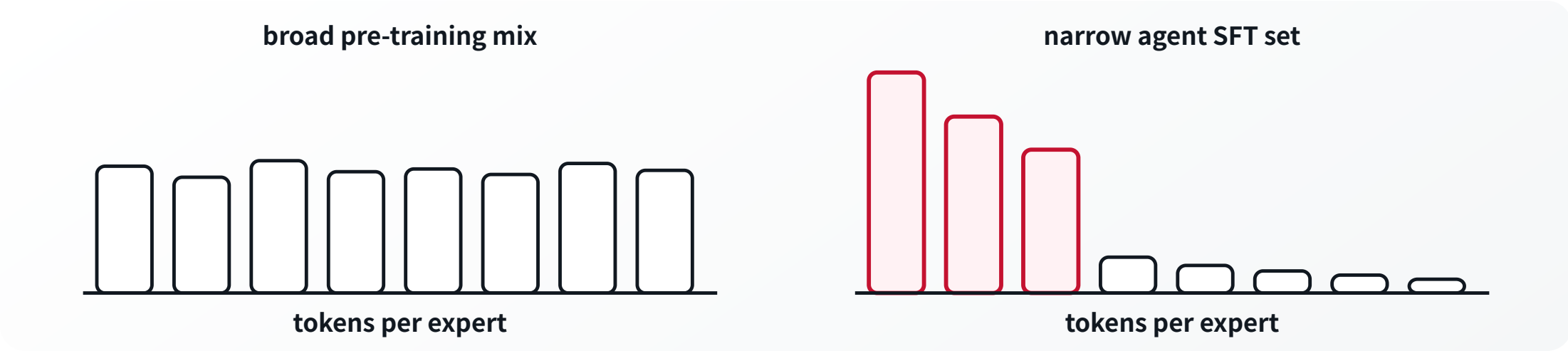
Running the training

Real SFT configurations

	SEQUENCE	BATCH	LEARNING RATE
Nemotron 3 Ultra	packed 294,912 then 515,000	64	1.5e-5 → 1e-6
MAI self-distillation	packed 128k	2,048	1.7e-5 → 5.2e-6, 2% warmup
K2 Horizon	512K, three phases	—	decayed on a high-quality subset

Nemotron 3 Ultra, § 3.1 ↗ MAI-Thinking-1, § 3.1.4 and § 3.5 ↗ K2 Horizon model card ↗

Expert routing during SFT

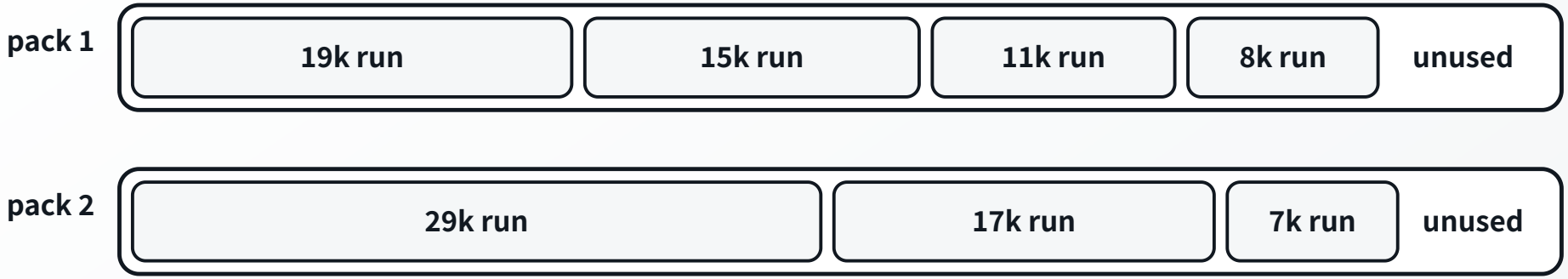


- The frontier models in this lecture are mixtures of experts. The models the public papers fine-tune are dense.
- Fine-tune an MoE on narrow agent data and a few experts take most of the tokens, like the right panel.
- MAI's fix: balance the routing hard during the supervised stage, a thousand times harder than during RL, and raise dropout to 0.15.

Microsoft AI, 2026 · MAI-Thinking-1, § 3.1.5 ↗

Packing whole conversations

Training runs on fixed-length sequences. Packing fills each one with several conversations, instead of padding most of it away.



- Each conversation is assigned to the pack whose remaining capacity it most tightly fits.
- No conversation is split or truncated, which is a deliberate choice against hallucination.
- Identical prompts are kept out of the same pack, and packs are shuffled afterwards.

NVIDIA, 2026 · Nemotron 3 Ultra, § 3.1.2 ↗

Template drift

- The template you train with has to be the template you serve with, down to the whitespace.
- Render the same trajectory twice, once with a tool message after it and once without, and you can get two different strings.
- When a tool message is appended, a conditional thinking block changes the earlier assistant turn, and the prefix no longer matches.

"The chat template should always match the format the model was trained with."

Transformers · writing chat templates ↗ TRL · training chat templates ↗

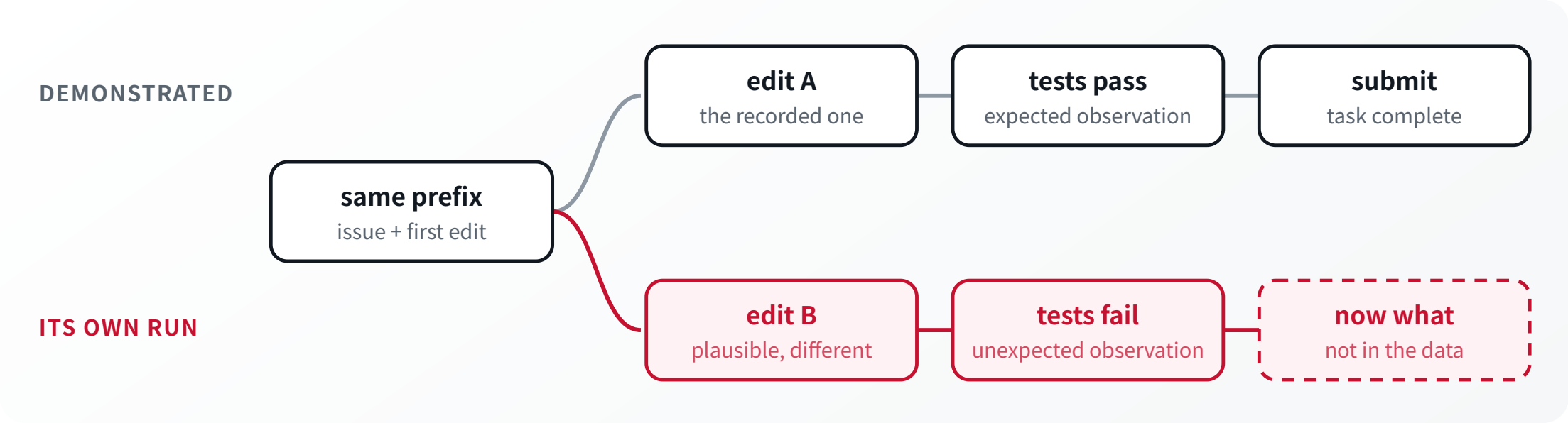
Released training artifacts

- K2 Horizon's model card has a stage-by-stage training table: steps, tokens, sequence length, and what each phase was for. That table is where the numbers on the configurations slide came from.
- It also links the Weights & Biases run, so you can look at the actual loss curves, and it ships intermediate checkpoints for every stage.
- Nemotron released the post-trained checkpoints together with the training data and the recipe.
- Reading one of these teaches a lot.

Nemotron 3 Ultra ↗ K2 Horizon ↗

Knowing it worked

The mismatch between training and running



- Training grades one action at a time, always continuing the recorded run.
- Deployed, there is no recording. Every action the model takes changes what it sees next.
- One different edit and it is in a state no training run ever visited. Low loss promised nothing about what happens there.

Ross et al., 2011 · DAgger ↗

Evaluation in the target harness

- Before trusting any number, check that you can overfit twenty examples. If the loss will not drop, the bug is upstream.
- Then evaluate by running the model in a harness, on tasks with checkable outcomes.
- Read published numbers carefully. Sometimes they are the best of several harnesses.
- Match the split to the claim. Keep all runs of one task on the same side, and split by repository if you want to say it generalizes to new ones.

OpenThoughts-Agent, Table 1 ↗ SWE-Gym, § 5.2 ↗

Robustness across harnesses

- Trajectories collected in one system teach that system's conventions along with the task.
- Collecting across several harnesses is how labs avoid training a model that only works in one.

NVIDIA, 2026 · Nemotron 3 Ultra, § 3.1.1 ↗

Regressions outside the target domain

- Train on one capability and you move the others, whether or not you measure it.
- Your mixture is the control here, and balancing it by examples is not the same as balancing it by tokens.
- So evaluate the things you were not trying to improve, too.

Microsoft AI, 2026 · MAI-Thinking-1, § 3.5 ↗

Handing off to RL

How much SFT before RL

Rank	Model (training recipe)	Benchmarks			Average	
		SWE-bench Verified (100)	OT-TBLite	Terminal-Bench 2.0	Raw	Normalized
1	OT-Agent-ColdSFT+RL-8B	**35.7** 1.8	**16.0** 1.6	**13.5** 1.5	**21.7**	**+1.2**
2	OT-Agent-SFT-8B (10K)	24.3 1.9	15.6 1.5	7.9 1.4	15.9	+0.5
3	OT-Agent-ColdSFT	23.7 1.6	14.8 1.4	6.7 1.1	15.1	+0.4
4	RL on Qwen3-8B (no SFT)	1.0 0.5	7.8 1.1	1.9 0.7	3.6	-0.9
5	Qwen3-8B	5.3 1.1	1.3 0.3	1.5 0.4	2.7	-1.1

- RL applied to no SFT at all barely moves the base model.
- SFT alone stops well short of what the pair reaches.
- The winning recipe starts RL from a checkpoint that was deliberately not trained to its own best score.

"RL provides the most gains when the SFT model is selected with RL in mind."

OpenThoughts-Agent, Table 11 ↗ MAI-Thinking-1, § 3.1.4 ↗

Questions

Next class · Training 2: Reinforcement Learning Basics

Backup · Exact masking flags

FRAMEWORK	FLAG	BEHAVIOR
TRL	`assistant_only_loss=True`	needs `{% generation %}` markers; known families get patched templates
TRL	`completion_only_loss`	prompt–completion datasets; on by default there
Axolotl	`roles_to_train: ["assistant"]`	default; loss on the listed roles only
Axolotl	`train_on_eos: turn`	the EOS of each trained turn is in the loss
LLaMA-Factory	`train_on_prompt: false`	default; prompt tokens masked
LLaMA-Factory	`mask_history: true`	last turn only; truncation keeps the final turn