EnvLunch: THE ENVIRONMENT AS A FREE LUNCH FOR LEARNING TO INTERACT

Anonymous authors
Paper under double-blind review

ABSTRACT

For agentic data analysis, language-model agents often require a large number of annotated tasks to learn how to interact effectively with executable environments (i.e., databases). While high-quality annotated tasks require heavy manual effort, an executable environment already provides rich interaction structure and verifiable outcomes. In this sense, the environment itself offers a free lunch, an additional set of verifiable tasks that require no human annotation. We propose *EnvLunch*, a warm-up stage that turns this environmental signal into agent training without human-written or model-generated task descriptions, before downstream agentic reinforcement learning. *EnvLunch* samples valid action trajectories, executes them to obtain final observations, and trains an agent to recover a functionally equivalent trajectory given only its final observation. This label-free warm-up enables the model to outperform the baseline by over 9% on average across all downstream training budgets, and with few labels, the baseline needs 8 to 32 times as many annotated examples to match the accuracy that the *EnvLunch* model reaches at its smallest training budget. Letting the environment alone define the task yields a larger gain than having the model write its own tasks over the same environments, and this warm-up also improves fine-tuning on benchmarks beyond data analysis. *EnvLunch* therefore saves annotated tasks and provides a better starting point for fine-tuning agentic models when such tasks are limited.

1 INTRODUCTION

Language model agents can interact with databases to perform long-horizon and complex data querying, analysis, and report generation (Zhang et al., 2024; Grijalba et al., 2024; Wu et al., 2025; Lei et al., 2025), a setting we refer to as **agentic data analysis**. Training agents to acquire such capabilities currently relies heavily on either large-scale collections of human-annotated verifiable tasks for reinforcement learning (Bai et al., 2024; Jin et al., 2025; Xi et al., 2026; Ma et al., 2025; Zhang et al., 2026a), or long-horizon trajectories distilled from much larger models (Sun et al., 2025; Su et al., 2025; Tongyi DeepResearch Team, 2025). Both approaches require substantial human effort and careful data curation (Huang et al., 2025).

Yet agentic training has an abundant source of supervision that is often underused, which is the **executable environment itself** (Yu et al., 2018; Li et al., 2023; Lei et al., 2025). In data analysis, databases and tables expose rich state and action spaces, while execution provides an automatic signal for the outcome of an action sequence (Tran et al., 2009; 2014; Wang et al., 2017; Silver & Sutton, 2025; Yao, 2025; Xi et al., 2025; Liu et al., 2026b; Meta PyTorch and Hugging Face, 2026). Existing methods partly reduce annotation costs by synthesising tasks with LLMs or human-defined templates, but still require these environments to be converted into explicit questions or task statements (Sun et al., 2025; Zhou et al., 2025; Stojanovski et al., 2025). We ask whether this step is necessary at all: can an agent learn to interact effectively by learning directly from information inherent in the environment? Any executable trajectory in an environment could expose the interaction process, providing signals for interaction behaviours such as recovering from failures or exploring the environment over long-horizon interactions. Such trajectories form an additional set of verifiable tasks without human annotation, which we call the free lunch of the environment.

Therefore, we propose *EnvLunch*, an environment-driven warm-up stage that runs before downstream agentic reinforcement learning. *EnvLunch* randomly composes valid actions into an exe-

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.

cutable trajectory, executes it to obtain a final observation, and trains the agent to recover a trajectory that could have produced that observation. Intuitively, this is like placing a target at a random location in a maze and training the agent to navigate to it: the target need not correspond to a meaningful downstream question, as long as reaching it requires effective interaction with the environment. To prevent the agent from simply copying the target observation, we create multiple perturbed copies of the original environment and execute the recovered trajectory in each, and agreement across these counterfactual environments determines the reward. *EnvLunch* requires no human or model annotation, is fully verifiable, and is lightweight, taking 20 minutes on CPU to construct 1600 warm-up tasks compared with more than 4 hours for model-based task synthesis.

We validate *EnvLunch* on Qwen3.5-4B and Qwen3.5-9B across six agentic data-analysis tasks. Averaged across different training budgets, *EnvLunch* improves accuracy by 9.6% at 4B and 9.3% at 9B, with the largest gains when labels are scarce, where the baseline requires 8–32$\times$ as many labelled examples to match the accuracy reached by the *EnvLunch* model at its smallest training budget. Environment-driven warm-up also yields larger gains than reinforcement learning on tasks that the model synthesises over the same environments. Across seven additional agentic tasks spanning text-to-SQL, question answering, software use, and embodied interaction, *EnvLunch* also improves downstream performance, despite using warm-up environments disjoint from the evaluation tasks. Finally, in a constructed environment where uncertainty can be computed exactly, *EnvLunch* produces the largest reduction in uncertainty about the environment, providing evidence that learning to interact contributes to the downstream gains.

Our contributions are threefold. First, we show that an executable environment alone can define verifiable agentic tasks without any annotation. Second, we propose *EnvLunch*, a warm-up stage for agents to learn from the environment without human annotation. Third, across two model scales and thirteen benchmarks, we show that *EnvLunch* substantially reduces the annotated examples needed for downstream agentic reinforcement learning, which suggests that executable environments contain useful training signals that can be extracted without first specifying the meaningful tasks. *EnvLunch* provides a preliminary step towards agentic training that learns directly from environments.

2 Related work

Learning from executable environments. Rewarding a model by executing its output originated in semantic parsing and program synthesis (Zhong et al., 2017; Le et al., 2022; Gehring et al., 2025) and now extends to agents operating over web pages, devices, and more (Qi et al., 2025; Bai et al., 2024; Jin et al., 2025; Wei et al., 2025; Pan et al., 2025; Qian et al., 2025; Zhang et al., 2026a). Executable environments thus act as verifiers that require neither rubrics nor annotation, which has motivated a shift from learning on text towards learning from experience (Silver & Sutton, 2025; Yao, 2025), RL infrastructure for existing environments (Xi et al., 2025; 2026; Liu et al., 2026b; Meta PyTorch and Hugging Face, 2026), and pipelines that synthesise new ones (Song et al., 2026; Xu et al., 2026; Dong et al., 2026; Tu et al., 2026; Wang et al., 2026a; Gandhi et al., 2026; Hu et al., 2025; Guo et al., 2025; Huang et al., 2025), including hand-crafted, model-generated or mined from the web (Zeng et al., 2026; Shi et al., 2026; Fang et al., 2026; Cai et al., 2025; Zhang et al., 2026c; Verma et al., 2025). *EnvLunch* builds on this view and carries it one step further: an environment that already supplies the verdict can also supply the task. Any environment these pipelines produce can therefore feed *EnvLunch* directly. A growing body of work lets the environment shape the tasks as well. Agents explore an environment and turn the resulting trajectory into an instruction (Sun et al., 2025; Su et al., 2025; Murty et al., 2024; Wang et al., 2026b; Mai et al., 2025; Yang et al., 2026), a proposer and a solver improve through self-play (Zhao et al., 2025a; Zhou et al., 2025; Zhai et al., 2025; Lu et al., 2026; Huang et al., 2026), task generators are paired with verifiers (Stojanovski et al., 2025; Li et al., 2025b; Chen et al., 2025b; Sullivan et al., 2025; Yang et al., 2025; Jain et al., 2025), and agents learn from the states their own actions reach along expert demonstrations (Zhang et al., 2026b). Together these works show that interaction with an environment is a rich source of training signal when little is annotated. *EnvLunch* follows this direction to the point where the environment defines the task and judges it, so no question needs to be written, and even a trajectory with no semantic meaning trains the multi-turn interaction behaviours that downstream tasks require.

Recovering a query from its output. Recovering a query from its output is a classic problem in databases (Tran et al., 2009; 2014; Zhang et al., 2013; Tan et al., 2018; Wang et al., 2017; Take-

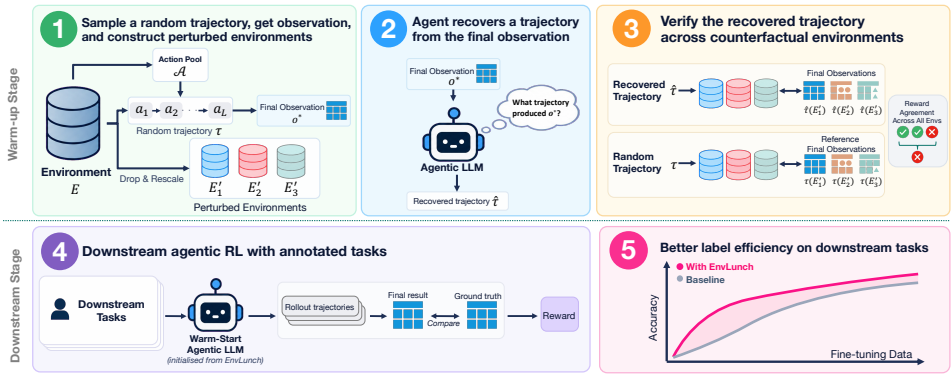

Figure 1: The overview of *EnvLunch*. In stages 1–3, *EnvLunch* performs warm-up by training the agent to recover a random synthesised trajectory in the environment. Stages 4–5 are ordinary downstream agentic RL, which then needs fewer annotated tasks.

nouchi et al., 2021; Orvalho et al., 2020; Pradhan & Haritsa, 2026), which establishes that an output constrains the program that produced it. This research identifies *instance-equivalent* programs, which agree on one database but compute different functions, and separates them with equivalence provers (He et al., 2024), language model judges (Zhao et al., 2025b), or additional databases designed to distinguish candidate programs (Chandra et al., 2015; Chen et al., 2025a; Klopfenstein et al., 2026). Execution-based text-to-SQL training meets the same issue as false positives of single-instance matching (Ma et al., 2025; Pourreza et al., 2025; Yao et al., 2026; Zhu et al., 2026), and synthetic-query training sidesteps it by providing the query explicitly (Liu et al., 2022; Yu et al., 2021; Li et al., 2025a; Zhang et al., 2025). *EnvLunch* turns this recovery problem into a training task for agents. Any trajectory that reproduces the target counts as a solution, which makes a task with no statement trainable, and the perturbed copies of Section 3 rule out degenerate solutions such as copying the output. Hindsight relabelling in goal-conditioned reinforcement learning shows that reached outcomes make effective goals (Andrychowicz et al., 2017; Ghosh et al., 2021). *EnvLunch* brings this principle to language-model agents and uses the outcomes of trajectories the agent never sees as the tasks themselves to bring the pressure of environment interaction.

3 METHOD

The free lunch in *EnvLunch* is the additional set of verifiable tasks that the environment provides without human annotation. Any executable trajectory can define a task, and its execution result is the answer, so neither a question nor a manually annotated label is needed. In this sense, the environment itself provides the task where agentic reinforcement learning can be performed. For agentic data analysis, the database serves as the environment, while database queries and computation statements constitute the executable action space. By constructing an action pool and performing online trajectory synthesis and filtering, and training the agent through counterfactual multi-environment verification (Figure 1), *EnvLunch* provides a fully automated warm-up for agentic capabilities.

Action Pool Construction. We define the action space associated with each environment as $\mathcal{A}$, e.g., `select`, `where`, and `group by` in data analysis. These operators form a generic interface that applies to any relational database without per-environment design. They are further organised into different levels according to the number of operators composed (see Appendix C). The actions can be composed sequentially and hierarchically into a trajectory. We write E for the environment, consisting of a database instance together with its tables, and $\mathcal{A}(o)$ for the actions that are valid given an observation o. Action parameters are instantiated dynamically from each environment; for example, the selected fields or number of retrieved rows are sampled according to the underlying database instance. Let $E \sim \mathcal{D}_E$ denote an environment sampled from the environment distribution, and let o_0 be the initial observation it exposes. Given E, *EnvLunch* composes a trajectory by repeatedly drawing a valid action and executing it:

$$\tau = (a_1, \ldots, a_L), \qquad a_\ell \in \mathcal{A}(o_{\ell-1}), \qquad o_\ell = a_\ell(o_{\ell-1}), \tag{1}$$

where each a_ℓ is drawn uniformly at random from the actions that are valid under the current observation, and $a_\ell(o_{\ell-1})$ denotes the observation obtained by executing a_ℓ on $o_{\ell-1}$. Since every action is drawn from those valid on the observation it will be applied to, τ is executable by construction. The length L is drawn from the difficulty levels of the pool and cut short whenever no valid action remains (Appendix C). For example, an *aggregation* or *join* action applied to $o_{\ell-1}$ yields a new observation o_ℓ. Executing the full trajectory produces the target observation $o^\star = o_L = \tau(E)$. *EnvLunch* reveals only E and $o^\star$, while hiding the sampled trajectory τ. The warm-up task is therefore

$$\mathcal{T}(E, \tau) = (E, o^\star). \tag{2}$$

The agent is asked to interact with E to discover the trajectory that could reproduce the target,

$$\hat{\tau} \in [\tau]_E := \{\tau' : \tau'(E) = o^\star\}, \tag{3}$$

so that the supervision is the execution result rather than a match against the hidden τ. Because the map from trajectories to observations is many-to-one, this class contains functionally equivalent trajectories that the synthesiser never sampled, and the counterfactual verification below is what removes its degenerate members. Note that recovering a short synthetic trajectory may require substantially longer interaction with the environment. A constructed trajectory of length $L = 5$, for example, may require many more turns of trial and reasoning than its length, providing substantially more opportunities for learning (Appendix J).

Online Trajectory Synthesis and Filtering. Each time a new action is appended to the constructed trajectory, we execute the partial trajectory in the environment to obtain o_ℓ, and rebuild the candidate pool $\mathcal{A}(o_\ell)$ for the next step, keeping an action only if it is executable on the current observation and can still change it (see Appendix C). For example, once the current execution has left only a handful of rows, a top-k ranking action no longer changes the observation and is therefore removed from the pool. Such synthetic trajectories need not resemble useful user tasks, for example, *"Pair every postseason batting record of a player with his every postseason fielding record, then group those pairs by the fielding error count and report how many pairs fall in each group."* Such tasks may have little practical meaning, but they still require the agent to interact with the environment and reason over intermediate observations to recover an executable trajectory. In addition to dynamically adjusting the candidate action pool, we also check whether the trajectory produces different results when executed under counterfactual perturbations of the environment. We retain a trajectory only if it executes and yields a non-degenerate result in every perturbed copy, and its result changes in at least one copy, so that an answer which merely copies the target observation cannot obtain the full reward. This rules out the reward hacking illustrated in Figure 6(a).

Counterfactual Multi-Environment Verification. After obtaining a synthetic trajectory and its final execution result, we use standard agentic RL to train the agent to recover a functionally equivalent trajectory. The agent is provided with the final execution result, the corresponding database instance, SQL and Python tools, and a minimal ReAct harness (Yao et al., 2023). Let $\hat{\tau}$ denote the trajectory produced by the agent, and $\hat{\tau}(E)$ its execution result in environment E, following the notation of Equation 1. We do not reward exact recovery of the original trajectory, because different trajectories can represent the same logic. Instead, we compare the execution results of the two trajectories. However, this setup is vulnerable to a simple form of reward hacking, where the agent could simply print the target observation result rather than reconstructing the trajectory. To address this, *EnvLunch* adopts counterfactual multi-environment verification. We perturb the original database by deleting rows and rescaling numeric columns (see Appendix C), to construct K parallel "what-if" databases $E'_1, \ldots, E'_K$, which the agent never sees. This replaces the equivalence class of Equation 3 by its intersection over the copies, $\bigcap_{k=1}^{K} \{\tau' : \tau'(E'_k) = \tau(E'_k)\}$, which a trajectory that merely prints $o^\star$ no longer belongs to. Without the copies, the agent hacks the reward and stops querying the database, all of its recovered queries become hardcoded constants, and accuracy on held-out warm-up tasks falls below that of the untrained model (Appendix C). We execute both trajectories in each copy and reward agreement across all perturbations:

$$R(\hat{\tau}; \tau, E'_{1:K}) \;=\; \Big(1 + w\,(1 - t/t_{\max})\Big) \;\cdot\; \min_{1 \le k \le K} m(\hat{\tau}(E'_k),\, \tau(E'_k)), \tag{4}$$

where $m \in [0, 1]$ measures agreement between two execution results and equals 1 only when they agree exactly (Appendix C gives its detailed form). The leading factor rewards brevity: t is the

number of turns the agent spends interacting with the environment before submitting $\hat{\tau}$, $t_{\max}$ is the turn budget of the harness, and $w \geq 0$ weights this bonus. Note that t counts the agent's own interaction and is typically far larger than the length L of the hidden trajectory τ. The min operation requires the recovered trajectory to agree with all K counterfactual environments (see Appendix C for more details on all hyperparameters).

Formally, given the environment E and the target observation $o^\star$, the agent interacts with the environment for multiple turns in a rollout and submits a recovered trajectory $\hat{\tau}$ in its final action

$$\hat{\tau} \sim \pi_\theta(\cdot \mid E, o^\star). \tag{5}$$

The *EnvLunch* objective is to maximise the expected reward of the recovered trajectory:

$$\max_\theta \; \mathbb{E}_{\substack{E\sim\mathcal{D}_E,\ \tau\sim\mathrm{Synth}(E),\\ \hat{\tau}\sim\pi_\theta(\cdot|E,o^\star)}} \left[R(\hat{\tau};\tau,E'_{1:K})\right], \tag{6}$$

where $\mathcal{D}_E$ denotes the distribution of environments that could be used for *EnvLunch* warm-up, $\mathrm{Synth}(E)$ is the randomised synthesiser of Equation 1, $o^\star = \tau(E)$ is the target observation it produces, and $E'_1, \ldots, E'_K$ are counterfactual perturbations of the original environment. This objective encourages the agent to recover not necessarily the original action sequence, but a trajectory that implements the same computation under changes to the environment. We optimise the model using standard GRPO (Shao et al., 2024).

4 Experimental setup

We evaluate *EnvLunch* on Qwen3.5-4B and Qwen3.5-9B (Qwen Team, 2026). For each backbone, *EnvLunch* uses the same pool of environments and warms up only once for every downstream benchmark. For each benchmark, we fine-tune both the released backbone checkpoint, which we call the **baseline**, and the corresponding *EnvLunch*-warmed checkpoint, which we call the ***EnvLunch* model**, using GRPO under the benchmark's native multi-turn interaction protocol. We leave its action space, turn limit, trajectory token budget, and official scorer unchanged. Each downstream fine-tuning setting is repeated with three independent random seeds. We plot downstream accuracy against the number of annotated training examples, evaluating the checkpoints at which the number of training examples seen is a power of two, to evaluate whether *EnvLunch* can improve downstream label efficiency (Appendix A gives the full *EnvLunch* and fine-tuning configurations). *EnvLunch* uses two environment corpora. The first is a relational corpus built on 112 databases from the training splits of Spider (Yu et al., 2018), BIRD (Li et al., 2023) and Archer (Zheng et al., 2024a), on which the agent interacts through SQL. The second is a dataframe corpus containing 398 tables from DataMind (Qiao et al., 2026), BIRD and DABench (Hu et al., 2024a), on which the agent interacts through Python. All warm-up tasks are built on these two corpora (Appendix C). The environments are not constructed for any particular downstream benchmark, and no evaluation environment from any benchmark is included in the warm-up data (Appendix C). Since *EnvLunch* needs no annotation, it can also run on the environments of a target benchmark, but we keep the two disjoint so that the downstream gains measure what transfers across environments rather than familiarity with the evaluation ones.

LLMs following the warm-up stage using *EnvLunch* are then further fine-tuned and evaluated on six downstream data-analysis benchmarks. Four present tables either alone or interleaved with text (ConvFinQA, Chen et al., 2022; TAT-QA, Zhu et al., 2021; MultiHiertt, Zhao et al., 2022; TableBench, Wu et al., 2025). The remaining two, DSEval (Zhang et al., 2024) and DataBench (Grijalba et al., 2024), require interacting with dataframes through code. Each keeps its own official test set and metric. The fine-tuning pool of a benchmark is drawn from its own training split and subsampled to at most 2048 examples. For benchmarks without a usable training split, we construct separate training and test partitions (Appendix D). We follow the official settings as closely as possible and add no extra harness design or result ensembling for chasing SOTA, since the quantity of interest is downstream label efficiency. The warm-up runs 100 steps of 16 tasks each, which takes about 4 hours for the 9B model and 3 hours for the 4B model. The warm-up needs no annotated tasks, runs once for each backbone and serves all downstream benchmarks. We also conduct experiments to rule out the possibility that the observed effect simply results from additional training on synthetic data (Section 6).

Following standard practice in self-supervised pre-training, we evaluate the warm-up stage by measuring the labelled downstream data required to reach a given performance level (Hénaff et al., 2020;

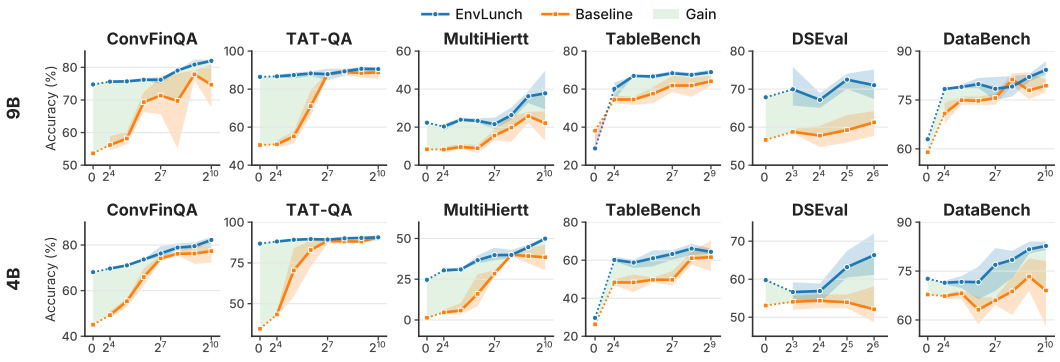

Figure 2: Label-efficiency curves for agentic data analysis. The top row shows the 9B model and the bottom row the 4B model, with one benchmark per column. The x-axis is the number of annotated examples on a log scale, and 0 denotes evaluation without downstream fine-tuning. Lines show the mean over three runs, and shaded bands span the corresponding min–max. The green areas between the lines are the gain from the warm-up; red regions indicate the reverse.

Chen et al., 2020a;b). We define downstream label efficiency as the number n of annotated examples required to reach a given accuracy. For each label-efficiency curve, we report two statistics. The *curve-level gain* is the mean accuracy difference between *EnvLunch* and the baseline across all training-data budgets and serves as our primary comparison. The *matching budget* is the smallest baseline budget needed to match or exceed *EnvLunch*'s accuracy, expressed as a multiple of the corresponding *EnvLunch* budget. It summarises the low-budget regime; "never ($> k\times$)" indicates that the baseline does not reach the target within the evaluated range.

5 Results

5.1 Agentic data analysis

Figure 2 gives an overview of the downstream fine-tuning curves, and Table 1 reports the detailed curve-level gain and matching budget. Each curve averages three independent runs, with shaded bands showing their minimum and maximum across runs. Across two models and six benchmarks, *EnvLunch* enables agentic LLMs to reach or even surpass baseline performance with substantially fewer annotated training examples. The advantage is most pronounced when labels are scarce. Notably, the warm-up tasks differ in form from the downstream ones. The warm-up trains the model to reconstruct synthetic trajectories, whereas the downstream tasks span data science coding, report analysis, table question answering, and other related data analysis tasks. Nevertheless, the performance gap generally persists as fine-tuning proceeds. On TableBench at 9B, although the *EnvLunch* model starts below the baseline, it overtakes the baseline after only 16 examples. In terms of label efficiency, on every curve, the baseline requires more annotated examples to match the accuracy achieved by the *EnvLunch* model at its smallest training budget. Specifically, the baseline requires between $8\times$ and $32\times$ as many examples when it reaches the target, with a median of $16\times$; on the two DSEval curves, it does not reach the target within the evaluated range.

Recent RL studies show that repeatedly training on a single example can keep improving agentic LLMs (Wang et al., 2025; Li et al., 2026), as a well-designed example leads the model through different intermediate states across iterations (Fu et al., 2026; Liu et al., 2026a). We test this extreme low-label regime and hold one annotated example fixed for both models and train on it for up to 32 steps, using examples at the 25th, 50th and 75th percentile of the per-group accuracy of the baseline (Appendix E). Figure 3 reports the accuracy difference after each step, averaged over the six benchmarks. After 32 steps on the same example, the *EnvLunch* model still leads by 5.5% to 12.6%, and the average difference stays positive at every step and every difficulty.

Table 1: Results on six agentic data-analysis benchmarks. *EnvLunch*/*Baseline* denote the mean accuracy averaged across all training budgets on each curve; Δ is their difference. Each value is the mean over three independent fine-tuning runs, with the standard deviation across runs shown as a subscript (Δ is computed by pairing runs). *Match* denotes the matching budget defined in Section 4.

		9B				4B			
Benchmark	Data substrate	*EnvLunch*	Baseline	Δ	Match	*EnvLunch*	Baseline	Δ	Match
ConvFinQA	Table + text	$77.9_{\pm 0.2}$	$68.2_{\pm 2.0}$	$\mathbf{+9.8}_{\pm 2.1}$	32×	$75.9_{\pm 0.6}$	$67.8_{\pm 2.2}$	$\mathbf{+8.1}_{\pm 1.8}$	8×
TAT-QA	Table + text	$88.7_{\pm 1.3}$	$75.9_{\pm 1.5}$	$\mathbf{+12.8}_{\pm 0.7}$	8×	$89.6_{\pm 0.3}$	$78.9_{\pm 1.3}$	$\mathbf{+10.7}_{\pm 1.4}$	8×
MultiHiertt	Table + text	$27.1_{\pm 3.1}$	$15.7_{\pm 0.7}$	$\mathbf{+11.4}_{\pm 3.1}$	32×	$38.9_{\pm 1.6}$	$24.7_{\pm 3.7}$	$\mathbf{+14.3}_{\pm 4.0}$	16×
TableBench	Table	$66.4_{\pm 1.4}$	$59.1_{\pm 2.5}$	$\mathbf{+7.3}_{\pm 3.6}$	8×	$62.3_{\pm 0.4}$	$53.1_{\pm 1.4}$	$\mathbf{+9.2}_{\pm 1.6}$	16×
DSEval	DataFrame	$70.2_{\pm 3.0}$	$59.3_{\pm 1.7}$	$\mathbf{+10.9}_{\pm 3.1}$	never	$60.8_{\pm 2.5}$	$53.6_{\pm 1.2}$	$\mathbf{+7.1}_{\pm 3.6}$	never
DataBench	DataFrame	$80.2_{\pm 1.2}$	$76.4_{\pm 1.1}$	$\mathbf{+3.8}_{\pm 2.0}$	16×	$76.4_{\pm 2.5}$	$68.0_{\pm 1.6}$	$\mathbf{+8.4}_{\pm 3.7}$	32×
Mean over 6 benchmarks				**+9.32**				**+9.63**	

Table 2: Seven benchmarks beyond data analysis. Δ is the curve-level gain, *Match* the matching budget of Section 4. Curves in Appendix D.1.

	9B		4B	
Benchmark	Δ	Match	Δ	Match
BIRD	**+4.8**	64×	**+6.8**	64×
Spider	**+2.2**	32×	**+1.7**	4×
Spider2-lite	**+4.6**	never	**+1.4**	1×
HotpotQA	**+15.3**	never	**+14.0**	32×
ALFWorld	**+3.7**	8×	**+2.1**	4×
AppWorld	**+5.7**	never	**+0.4**	1×
SpreadsheetBench	**+2.0**	never	**−0.2**	1×
Mean	**+5.48**		**+3.73**	

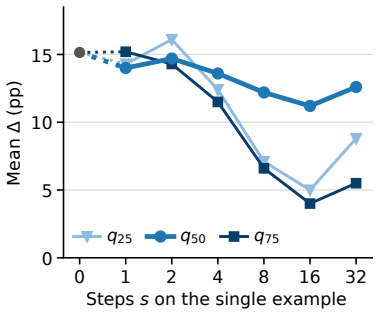

Figure 3: One labelled example, 9B. Mean Δ after s steps on a *single* fixed example; q_k: kth-percentile example. Details in Table A4.

5.2 Beyond agentic data analysis

We extend the *EnvLunch* model to seven downstream benchmarks beyond agentic data analysis (see details in Appendix D.1) including text-to-SQL (BIRD, Li et al., 2023; Spider, Yu et al., 2018; Spider2-lite, Lei et al., 2025), question answering (HotpotQA, Yang et al., 2018), embodied simulation (ALFWorld, Shridhar et al., 2021), application use (AppWorld, Trivedi et al., 2024), and spreadsheet editing (SpreadsheetBench, Ma et al., 2024). Table 2 reports the curve-level gain and the matching budget on these seven benchmarks, and Appendix D.1 plots the corresponding curves. For these tasks, the warm-up differs entirely from the downstream tasks in task formulation and domain, yet *EnvLunch* still exhibits a degree of generalisation, improving downstream label efficiency. Although the free lunch claimed by *EnvLunch* is in the form of additional data rather than reduced compute, across the data analysis and additional benchmarks, the extra fine-tuning required for the baseline to match *EnvLunch* at its smallest budget takes longer than the warm-up stage (Appendix B).

6 Analysis of the warm-up

We conduct a detailed analysis of the trajectories and design additional controlled experiments to understand what the model learns from the *EnvLunch* warm-up and how it shapes the model's behaviour, enabling it to make better use of downstream data. We first examine four intuitive alternative explanations, then analyse and quantify the characteristics of the trajectories under a fair setting, and finally measure how the model reduces its uncertainty about the environment in a controlled experimental setting.

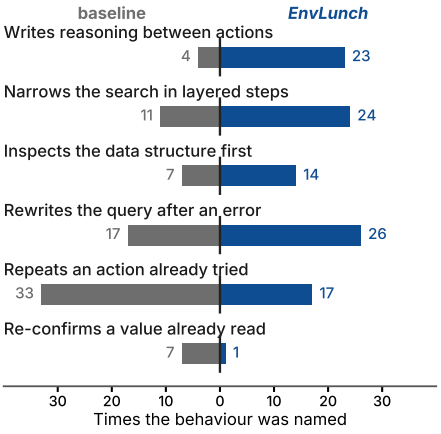

Figure 4: Behaviours identified in trajectories, where both models obtain the same score. See details in Appendix G.

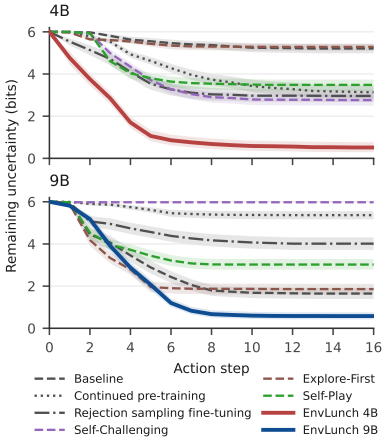

Figure 5: Remaining uncertainty about the environment after each action, with 95% intervals over tasks.

Alternative Explanations. *EnvLunch* does not ask the model to learn an explicit world model of the environment. Instead, the task simply requires the model to interact with and reason about the environment in order to reconstruct a randomly sampled trajectory. What, then, does the model actually learn during this RL process? We consider four intuitive possibilities. First, the model is unlikely to benefit simply by memorising the warm-up environments, since none of them appears in any evaluation set (Section 4), and benchmarks whose domains do not overlap with them, such as ALFWorld and AppWorld, still benefit from *EnvLunch*. Two other possibilities are that the model learns the schema of the warm-up environments, or that it imitates some key actions appearing in successful trajectories. We test these hypotheses by replacing the RL-based warm-up with continued pre-training on environment text and rejection sampling fine-tuning on successful trajectories, respectively. Neither of these two warm-up approaches yields significant gains (see Appendix F). A fourth possibility is that any reinforcement learning on these environments brings the gain, whatever the tasks are. We test it by keeping the *EnvLunch* stage unchanged and replacing only its task pool with tasks that the LLM synthesises itself over the same databases and tables, following three lines of prior work. In Self-Challenging (Zhou et al., 2025), the model inspects an environment and writes a question together with a program that answers it as verification. In Explore-First (Sun et al., 2025; Su et al., 2025), the model first interacts with the environment freely and then rewrites its trajectory into a question and an answer. In Self-Play (Zhao et al., 2025a; Huang et al., 2026), the model proposes a question and solves it eight times, taking the majority answer as the label. All three share with *EnvLunch* the harness and all other settings. At both scales, all three fall short of *EnvLunch* on the downstream curves (Appendix F.1). In each of them the model labels its own tasks, and nothing checks that the label answers the question it wrote, which likely makes the training signal less reliable. Detailed experimental settings are provided in Appendix F.

Trajectory-Level Behavioural Analysis. We randomly sample trajectories from the baseline model and the model after *EnvLunch* warm-up across all benchmarks, and retain only items on which both models obtain the same score, ensuring a fair comparison. We shuffle the order within each pair and ask Claude Sonnet 5 to examine the two trajectories and describe their behavioural differences. We then manually verify, categorise, and quantify the identified differences (see Appendix G for details). As shown in Figure 4, after *EnvLunch* warm-up, the model more often reasons explicitly at key intermediate steps, more frequently inspects the data structure before executing an operation, progressively narrows down the search space, and rewrites the query after encountering errors. In contrast, the baseline more frequently repeats previously attempted operations or reconfirms values that it has already retrieved. Figure 6 presents a concrete case study. In the *EnvLunch* stage in panel (a), the model observes that the values in the final observation are substantially smaller than the mean returned by its query. It therefore reasons further and ultimately obtains the correct answer. In

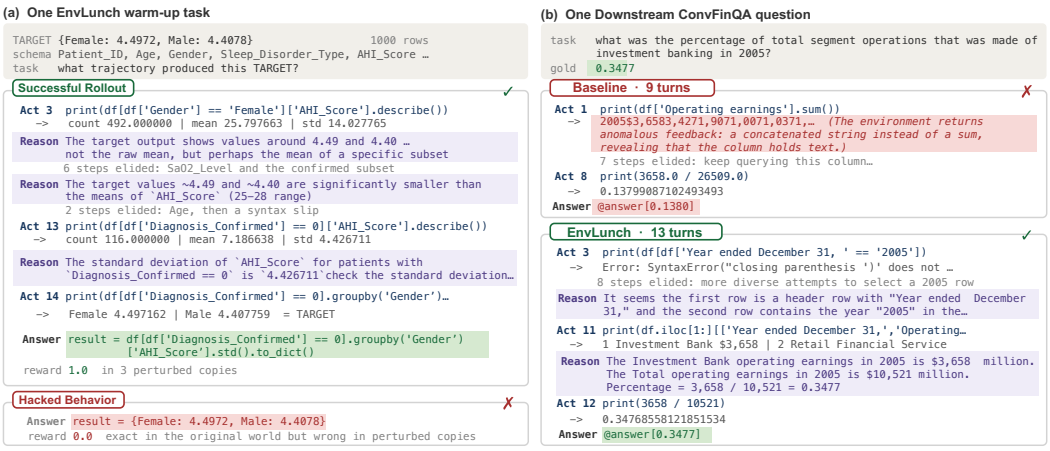

Figure 6: Two excerpted trajectories from the 9B models. **(a)** An *EnvLunch* warm-up task. **(b)** A ConvFinQA question after both models have been fine-tuned on the same labelled examples.

the downstream ConvFinQA task in panel (b), although the task type is different, the model exhibits a similar pattern of reasoning and consequently takes more appropriate actions. Next, we investigate what kind of environment interaction behaviour this reasoning shapes.

Uncertainty Reduction in a Controlled Environment. As shown by the trajectory analysis above, after warm-up, the model appears to learn how to interact with the environment to reduce uncertainty and narrow the search space for subsequent steps. However, real-world benchmark environments are large and complex, making it difficult to cleanly and accurately measure how well a model understands the environment. Following prior work that measures information seeking by how strongly an agent narrows hypotheses about a hidden state (Grand et al., 2024; Hu et al., 2024b; Golde et al., 2025), we construct a controlled environment with exactly measurable uncertainty. The model queries a constructed sales table whose three columns each admit one of four encodings, yielding 64 possible environments. All environments share the same prompt and final answer and differ only in their hidden encodings, which can be distinguished only through queries. Following possible-world semantics for incomplete databases (Imielinski & Witold Lipski, 1984), we execute each model action across all 64 hidden tables and discard those whose output differs from the model's observation. The remaining tables are consistent with the model's observations, so we define uncertainty as $\log_2$ of their number, decreasing from 6 to 0 bits as the encodings are identified (Appendix H). Figure 5 shows that *EnvLunch* reduces uncertainty more effectively than the baseline and the continued pre-training, rejection sampling fine-tuning and synthesised-task variants at both scales. This difference is not explained by taking more actions: at 4B, *EnvLunch* and rejection sampling fine-tuning take a similar number of actions but achieve substantially different uncertainty reduction. The warm-up therefore shapes the model to select actions that reduce uncertainty about the environment, providing a behavioural mechanism for the downstream gains in Section 5.

7 Conclusion

We propose *EnvLunch*, a warm-up stage that trains agentic LLMs to reconstruct randomly synthesised trajectories from the environment alone before downstream agentic reinforcement learning. *EnvLunch* improves accuracy by over 9% on average across six agentic data analysis benchmarks at both model scales. When labels are scarce, the baseline requires 8–$32\times$ as many annotated examples to reach comparable performance, and the gains transfer to almost all of the seven additional benchmarks. Our study focuses on executable environments with verifiable tasks, with limitations discussed in Appendix I. More broadly, these results show that an environment's existing information and executable dynamics can provide useful supervision for learning agent behaviour before any downstream question is written.

AI USE STATEMENT

We used Claude Sonnet 5 to describe behavioural differences between paired trajectories (Section 6); the authors checked every description against the trajectories before grouping and counting them. Language models were also used to assist with editing the text and writing code. The authors take full responsibility for the content of the paper.

ETHICS STATEMENT

This work trains and evaluates language model agents on publicly available benchmarks and on the databases those benchmarks already distribute. We collect no new data from people and run no studies with human subjects.

The warm-up stage executes model-written code and model-written SQL against these environments. All such execution is confined to sandboxed subprocesses with read-only database access, memory and wall-clock limits and a private working directory, both to protect the host and to leave the environments themselves unmodified.

One property of the method is also a risk. *EnvLunch* produces training tasks from an environment with no annotator in the loop, which makes it cheap, and it also means that whatever an environment contains propagates into the trained model without any step at which a person inspects the tasks. The filtering of Appendix C rejects trajectories that are degenerate or unverifiable, so it checks correctness and does not check content. Applying the method to an environment holding sensitive, proprietary or skewed records would carry those properties into the resulting agent, and a deployment in such a setting should add an inspection step that our pipeline does not provide.

REPRODUCIBILITY STATEMENT

Section 4 states the downstream protocol and Appendix D lists per-benchmark turn limits, evaluation sets and scorers; the two models on a curve differ only in their initial weights. The code, the corpus construction, the evaluation code and the scripts will be released.

REFERENCES

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems 30 (NeurIPS 2017)*, 2017.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. DigiRL: training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

Shihao Cai, Runnan Fang, Jialong Wu, Baixuan Li, Xinyu Wang, Yong Jiang, Liangcai Su, Liwen Zhang, Wenbiao Yin, Zhen Zhang, Fuli Feng, Pengjun Xie, and Xiaobin Wang. AutoForge: automated environment synthesis for agentic reinforcement learning. *arXiv preprint arXiv:2512.22857*, 2025.

Bikash Chandra, Bhupesh Chawda, Biplab Kar, K. V. Maheshwara Reddy, Shetal Shah, and S. Sudarshan. Data generation for testing and grading SQL queries. *The VLDB Journal*, 24(6):731–755, 2015.

Chunyu Chen, Zhengjie Miao, Yong Zhang, and Jiannan Wang. ParSEval: Plan-aware test database generation for SQL equivalence evaluation. *Proceedings of the VLDB Endowment*, 18(11):4750–4762, 2025a.

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025b.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020a.

Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey Hinton. Big self-supervised models are strong semi-supervised learners. In *Advances in Neural Information Processing Systems 33 (NeurIPS 2020)*, 2020b.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2022.

Guanting Dong, Junting Lu, Junjie Huang, Wanjun Zhong, Longxiang Liu, Shijue Huang, Zhenyu Li, Yang Zhao, Xiaoshuai Song, Xiaoxi Li, Jiajie Jin, Yutao Zhu, Hanbin Wang, Fangyu Lei, Qinyu Luo, Mingyang Chen, Zehui Chen, Jiazhan Feng, Ji-Rong Wen, and Zhicheng Dou. Agentworld: Scaling real-world environment synthesis for evolving general agent intelligence. *arXiv preprint arXiv:2604.18292*, 2026.

Runnan Fang, Shihao Cai, Baixuan Li, Jialong Wu, Guangyu Li, Wenbiao Yin, Xinyu Wang, Xiaobin Wang, Liangcai Su, Zhen Zhang, Shibin Wu, Zhengwei Tao, Yong Jiang, Pengjun Xie, Ningyu Zhang, Fei Huang, Wentao Zhang, and Jingren Zhou. Towards general agentic intelligence via environment scaling. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 17610–17621, 2026.

Zixuan Fu, Bingxiang He, Yuxin Zuo, Haohuan Huang, Jinqian Zhang, Ruhang Xiao, Cheng Qian, Qinyu Luo, Huan-ang Gao, Yudong Wang, et al. Rethinking on-policy distillation of large language models II: One training example. *arXiv preprint arXiv:2609.04172*, 2026.

Kanishk Gandhi, Shivam Garg, Noah D. Goodman, and Dimitris Papailiopoulos. Endless terminals: Scaling RL environments for terminal agents. *arXiv preprint arXiv:2601.16443*, 2026.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Quentin Carbonneaux, Taco Cohen, and Gabriel Synnaeve. RLEF: Grounding code LLMs in execution feedback with reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Dibya Ghosh, Abhishek Gupta, Ashwin Reddy, Justin Fu, Coline Devin, Benjamin Eysenbach, and Sergey Levine. Learning to reach goals via iterated supervised learning. In *International Conference on Learning Representations (ICLR 2021)*, 2021.

Jonas Golde, Patrick Haller, Fabio Barth, and Alan Akbik. MastermindEval: A simple but scalable reasoning benchmark. In *ICLR 2025 Workshop on Reasoning and Planning for Large Language Models*, 2025.

Gabriel Grand, Valerio Pepe, Jacob Andreas, and Joshua B. Tenenbaum. Loose LIPS sink ships: Asking questions in battleship with language-informed program sampling. In *Proceedings of the 46th Annual Meeting of the Cognitive Science Society (CogSci)*, 2024.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 13471–13488, 2024.

Jiacheng Guo, Ling Yang, Peter Chen, Qixin Xiao, Yinjie Wang, Xinzhe Juan, Jiahao Qiu, Ke Shen, and Mengdi Wang. GenEnv: difficulty-aligned co-evolution between LLM agents and environment simulators. *arXiv preprint arXiv:2512.19682*, 2025.

Yang He, Pinhan Zhao, Xinyu Wang, and Yuepeng Wang. VeriEQL: Bounded equivalence verification for complex SQL queries with integrity constraints. *Proceedings of the ACM on Programming Languages*, 8(OOPSLA1):1071–1099, 2024. doi: 10.1145/3649849.

Olivier J. Hénaff, Aravind Srinivas, Jeffrey De Fauw, Ali Razavi, Carl Doersch, S. M. Ali Eslami, and Aäron van den Oord. Data-efficient image recognition with contrastive predictive coding. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, 2020.

Mengkang Hu, Pu Zhao, Can Xu, Qingfeng Sun, Jianguang Lou, Qingwei Lin, Ping Luo, and Saravan Rajmohan. AgentGen: enhancing planning abilities for large language model based agent via environment and task generation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Volume 1 (KDD 2025)*, 2025.

Xueyu Hu, Ziyu Zhao, Shuang Wei, Ziwei Chai, Qianli Ma, Guoyin Wang, Xuwu Wang, Jing Su, Jingjing Xu, Ming Zhu, Yao Cheng, Jianbo Yuan, Jiwei Li, Kun Kuang, Yang Yang, Hongxia Yang, and Fei Wu. InfiAgent-DABench: Evaluating agents on data analysis tasks. In *International Conference on Machine Learning (ICML 2024)*, 2024a.

Zhiyuan Hu, Chumin Liu, Xidong Feng, Yilun Zhao, See-Kiong Ng, Anh Tuan Luu, Junxian He, Pang Wei Koh, and Bryan Hooi. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Chengsong Huang, Wenhao Yu, Xiaoyang Wang, Hongming Zhang, Zongxia Li, Ruosen Li, Jiaxin Huang, Haitao Mi, and Dong Yu. R-Zero: Self-evolving reasoning LLM from zero data. In *International Conference on Learning Representations (ICLR 2026)*, 2026.

Yuchen Huang, Sijia Li, Minghao Liu, Wei Liu, Shijue Huang, Zhiyuan Fan, Hou Pong Chan, and Yi R. Fung. Environment scaling for interactive agentic experience collection: A survey. *arXiv preprint arXiv:2511.09586*, 2025.

Tomasz Imielinski and Jr. Witold Lipski. Incomplete information in relational databases. *Journal of the ACM*, 31(4):761–791, 1984. doi: 10.1145/1634.1886.

Naman Jain, Jaskirat Singh, Manish Shetty, Liang Zheng, Koushik Sen, and Ion Stoica. R2E-Gym: procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Conference on Language Modeling (COLM 2025)*, 2025.

Bowen Jin, Hansi Zeng, Zhenrui Yue, Jinsung Yoon, Sercan Arik, Dong Wang, Hamed Zamani, and Jiawei Han. Search-R1: training LLMs to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516*, 2025.

Won Kim, Byoung-Ju Choi, Eui-Kyeong Hong, Soo-Kyung Kim, and Doheon Lee. A taxonomy of dirty data. *Data Mining and Knowledge Discovery*, 7(1):81–99, 2003. doi: 10.1023/A:1021564703268.

Rocky Klopfenstein, Yang He, Andrew Tremante, Yuepeng Wang, Nina Narodytska, and Haoze Wu. SpotIt: evaluating Text-to-SQL evaluation with formal verification. In *International Conference on Learning Representations (ICLR 2026)*, 2026.

Hung Le, Yue Wang, Akhilesh Deepak Gotmare, Silvio Savarese, and Steven C. H. Hoi. CodeRL: Mastering code generation through pretrained models and deep reinforcement learning. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, pp. 21314–21328, 2022.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-SQL workflows. In *International Conference on Learning Representations (ICLR 2025)*, 2025.

Haoyang Li, Shang Wu, Xiaokang Zhang, Xinmei Huang, Jing Zhang, Fuxin Jiang, Shuai Wang, Tieying Zhang, Jianjun Chen, Rui Shi, Hong Chen, and Cuiping Li. OmniSQL: Synthesizing high-quality text-to-SQL data at scale. *Proceedings of the VLDB Endowment*, 18, 2025a.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as A database interface? A BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems 36 (NeurIPS 2023)*, 2023.

Peiji Li, Jiasheng Ye, Yongkang Chen, Linyang Li, Yichuan Ma, Zijie Yu, Ganqu Cui, Haozhan Li, Jiacheng Chen, Chengqi Lyu, Wenwei Zhang, Qipeng Guo, Dahua Lin, Bowen Zhou, and Kai Chen. InternBootcamp: boosting LLM reasoning with verifiable task scaling. *arXiv preprint arXiv:2508.08636*, 2025b.

Yiyuan Li, Zhen Huang, Yanan Wu, Weixun Wang, Xuefeng Li, Yijia Luo, Wenbo Su, Bo Zheng, and Pengfei Liu. One sample to rule them all: Extreme data efficiency in multidiscipline reasoning with reinforcement learning. *arXiv preprint arXiv:2601.03111*, 2026.

Qian Liu, Bei Chen, Jiaqi Guo, Morteza Ziyadi, Zeqi Lin, Weizhu Chen, and Jian-Guang Lou. TAPEX: Table pre-training via learning a neural SQL executor. In *International Conference on Learning Representations (ICLR 2022)*, 2022.

Wei Liu, Siya Qi, Yali Du, and Yulan He. Position: Self-play only evolves when self-synthetic pipeline ensures learnable information gain. In *Forty-third International Conference on Machine Learning Position Paper Track*, 2026a.

Zichen Liu, Anya Sims, Keyu Duan, Changyu Chen, Simon Yu, Xiangxin Zhou, Haotian Xu, Shaopan Xiong, Bo Liu, Chenmien Tan, Chuen Yang Beh, Weixun Wang, Hao Zhu, Weiyan Shi, Diyi Yang, Michael Shieh, Yee Whye Teh, Wee Sun Lee, and Min Lin. GEM: a gym for agentic LLMs. In *International Conference on Learning Representations (ICLR 2026)*, 2026b.

Hongliang Lu, Yuhang Wen, Pengyu Cheng, Ruijin Ding, Jiaqi Guo, Haotian Xu, Chutian Wang, Haonan Chen, Xiaoxi Jiang, and Guanjun Jiang. Search self-play: Pushing the frontier of agent capability without supervision. In *International Conference on Learning Representations (ICLR 2026)*, 2026.

Peixian Ma, Xialie Zhuang, Chengjin Xu, Xuhui Jiang, Ran Chen, and Jian Guo. SQL-R1: Training natural language to SQL reasoning model by reinforcement learning. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards challenging real world spreadsheet manipulation. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024.

Shinji Mai, Yunpeng Zhai, Ziqian Chen, Cheng Chen, Anni Zou, Shuchang Tao, Zhaoyang Liu, and Bolin Ding. CuES: A curiosity-driven and environment-grounded synthesis framework for agentic RL. *arXiv preprint arXiv:2512.01311*, 2025.

Meta PyTorch and Hugging Face. OpenEnv: an interface library for RL post-training with environments. GitHub repository and blog post, 2026. URL https://github.com/huggingface/OpenEnv.

Shikhar Murty, Hao Zhu, Dzmitry Bahdanau, and Christopher D. Manning. NNetNav: unsupervised learning of browser agents through environment interaction in the wild. *arXiv preprint arXiv:2410.02907*, 2024.

Pedro Orvalho, Miguel Terra-Neves, Miguel Ventura, Ruben Martins, and Vasco Manquinho. SQUARES: A SQL synthesizer using query reverse engineering. *Proceedings of the VLDB Endowment*, 13(12):2853–2856, 2020.

Jiayi Pan, Xingyao Wang, Graham Neubig, Navdeep Jaitly, Heng Ji, Alane Suhr, and Yizhe Zhang. Training software engineering agents and verifiers with SWE-Gym. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267, pp. 47717–47737, 2025.

Mohammadreza Pourreza, Shayan Talaei, Ruoxi Sun, Xingchen Wan, Hailong Li, Azalia Mirhoseini, Amin Saberi, and Sercan O. Arik. Reasoning-SQL: Reinforcement learning with SQL tailored partial rewards for reasoning-enhanced text-to-SQL. *arXiv preprint arXiv:2503.23157*, 2025.

Ahana Pradhan and Jayant Haritsa. XPOSE: Bi-directional engineering for hidden query extraction. In *Proceedings of the 37th International Conference on Database and Expert Systems Applications (DEXA 2026)*, volume 16860 of *Lecture Notes in Computer Science*, 2026.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. WebRL: training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations (ICLR 2025)*, 2025.

Cheng Qian, Emre Can Acikgoz, Qi He, Hongru Wang, Xiusi Chen, Dilek Hakkani-Tür, Gokhan Tur, and Heng Ji. ToolRL: reward is all tool learning needs. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Shuofei Qiao, Yanqiu Zhao, Zhisong Qiu, Xiaobin Wang, Jintian Zhang, Zhao Bin, Ningyu Zhang, Yong Jiang, Pengjun Xie, Fei Huang, and Huajun Chen. Scaling generalist data-analytic agents. In *International Conference on Learning Representations (ICLR 2026)*, 2026.

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

Erhard Rahm and Hong Hai Do. Data cleaning: Problems and current approaches. *IEEE Data Engineering Bulletin*, 23(4):3–13, 2000.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yucheng Shi, Zhenwen Liang, Kishan Panaganti, Dian Yu, Wenhao Yu, and Haitao Mi. Learning to build the environment: Self-evolving reasoning RL via verifiable environment synthesis. *arXiv preprint arXiv:2605.14392*, 2026.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. Megatron-LM: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*, 2019.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning text and embodied environments for interactive learning. In *International Conference on Learning Representations (ICLR 2021)*, 2021.

David Silver and Richard S. Sutton. Welcome to the era of experience. Google DeepMind; preprint of a chapter in *Designing an Intelligence*, MIT Press, 2025. URL https://storage.googleapis.com/deepmind-media/Era-of-Experience/The%20Era%20of%20Experience%20Paper.pdf.

Xiaoshuai Song, Haofei Chang, Guanting Dong, Yutao Zhu, Ji-Rong Wen, and Zhicheng Dou. EnvScaler: scaling tool-interactive environments for LLM agent via programmatic synthesis. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 8326–8357, 2026.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Hongjin Su, Ruoxi Sun, Jinsung Yoon, Pengcheng Yin, Tao Yu, and Sercan Ö. Arik. Learn-by-interact: A data-centric framework for self-adaptive agents in realistic environments. In *International Conference on Learning Representations (ICLR 2025)*, 2025.

Michael Sullivan, Mareike Hartmann, and Alexander Koller. Procedural environment generation for tool-use agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2025.

Qiushi Sun, Kanzhi Cheng, Zichen Ding, Chuanyang Jin, Yian Wang, Fangzhi Xu, Zhenyu Wu, Chengyou Jia, Liheng Chen, Zhoumianze Liu, Ben Kao, Guohao Li, Junxian He, Yu Qiao, and Zhiyong Wu. OS-Genesis: Automating GUI agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2025.

Keita Takenouchi, Takashi Ishio, Joji Okada, and Yuji Sakata. PATSQL: Efficient synthesis of SQL queries from example tables with quick inference of projected columns. *Proceedings of the VLDB Endowment*, 14(11):1937–1949, 2021.

Wei Chit Tan, Meihui Zhang, Hazem Elmeleegy, and Divesh Srivastava. REGAL+: reverse engineering SPJA queries. *Proceedings of the VLDB Endowment*, 11(12):1982–1985, 2018.

Tongyi DeepResearch Team. Tongyi DeepResearch technical report. *arXiv preprint arXiv:2510.24701*, 2025.

Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. Query by output. In *Proceedings of the 2009 ACM SIGMOD International Conference on Management of Data*, pp. 535–548, 2009.

Quoc Trung Tran, Chee-Yong Chan, and Srinivasan Parthasarathy. Query reverse engineering. *The VLDB Journal*, 23(5):721–746, 2014.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A controllable world of apps and people for benchmarking interactive coding agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. ScaleEnv: scaling environment synthesis from scratch for generalist interactive tool-use agent training. *arXiv preprint arXiv:2602.06820*, 2026.

Vivek Verma, David Huang, William Chen, Dan Klein, and Nicholas Tomlin. Measuring general intelligence with generated games. *arXiv preprint arXiv:2505.07215*, 2025.

Chenglong Wang, Alvin Cheung, and Rastislav Bodík. Synthesizing highly expressive SQL queries from input-output examples. In *Proceedings of the 38th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pp. 452–466, 2017.

Yiping Wang, Qing Yang, Zhiyuan Zeng, Liliang Ren, Liyuan Liu, Baolin Peng, Hao Cheng, Xuehai He, Kuan Wang, Jianfeng Gao, Weizhu Chen, Shuohang Wang, Simon Shaolei Du, and Yelong Shen. Reinforcement learning for reasoning in large language models with one training example. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent world model: Infinity synthetic environments for agentic reinforcement learning. In *International Conference on Machine Learning (ICML 2026)*, 2026a.

Ziyi Wang, Yuxuan Lu, Yimeng Zhang, Pei Chen, Ziwei Dong, Jing Huang, Jiri Gesi, Xianfeng Tang, Chen Luo, Qun Liu, Yisi Sang, Hanqing Lu, Manling Li, Jin Lai, and Dakuo Wang. Trajectory2Task: Training robust tool-calling agents with synthesized yet verifiable data for complex user intents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026b.

Yuxiang Wei, Olivier Duchenne, Jade Copet, Quentin Carbonneaux, Lingming Zhang, Daniel Fried, Gabriel Synnaeve, Rishabh Singh, and Sida I. Wang. SWE-RL: advancing LLM reasoning via reinforcement learning on open software evolution. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Xianjie Wu, Jian Yang, Linzheng Chai, Ge Zhang, Jiaheng Liu, Xinrun Du, Di Liang, Daixin Shu, Xianfu Cheng, Tianzhen Sun, Guanglin Niu, Tongliang Li, and Zhoujun Li. TableBench: A comprehensive and complex benchmark for table question answering. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25497–25506, 2025.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Lu Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. AgentGym: Evaluating and training large language model-based agents across diverse environments. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 27914–27961, 2025.

Zhiheng Xi, Jixuan Huang, Chenyang Liao, Baodai Huang, Jiaqi Liu, Honglin Guo, Yajie Yang, Rui Zheng, Junjie Ye, Jiazheng Zhang, Wenxiang Chen, Wei He, Yiwen Ding, Guanyu Li, Zehui Chen, Zhengyin Du, Xuesong Yao, Yufei Xu, Jiecao Chen, Tao Gui, Zuxuan Wu, Qi Zhang, Xuanjing Huang, and Yu-Gang Jiang. AgentGym-RL: an open-source framework to train LLM agents for long-horizon decision making via multi-turn RL. In *International Conference on Learning Representations (ICLR 2026)*, 2026.

Minrui Xu, Zilin Wang, Mengyi Deng, Zhiwei Li, Zhicheng Yang, Xiao Zhu, Yinhong Liu, Boyu Zhu, Baiyu Huang, Chao Chen, Heyuan Deng, Fei Mi, Lifeng Shang, Xingshan Zeng, and Zhijiang Guo. EnvFactory: scaling tool-use agents via executable environments synthesis and robust RL. *arXiv preprint arXiv:2605.18703*, 2026.

John Yang, Kilian Lieret, Carlos E. Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: scaling data for software engineering agents. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025, Datasets and Benchmarks Track)*, 2025.

Shidong Yang, Ziyu Ma, Tongwen Huang, Yiming Hu, Yong Wang, and Xiangxiang Chu. CoEvolve: training LLM agents via agent-data mutual evolution. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

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 *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Shunyu Yao. The second half. Personal blog, 2025. URL `https://ysymyth.github.io/The-Second-Half/`.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR 2023)*, 2023.

Zhewei Yao, Guoheng Sun, Łukasz Borchmann, Zheyu Shen, Minghang Deng, Bohan Zhai, Hao Zhang, Ang Li, and Yuxiong He. Arctic-Text2SQL-R1: Simple rewards, strong reasoning in text-to-SQL. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 26966–26995, 2026.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

Tao Yu, Chien-Sheng Wu, Xi Victoria Lin, Bailin Wang, Yi Chern Tan, Xinyi Yang, Dragomir Radev, Richard Socher, and Caiming Xiong. GraPPa: Grammar-augmented pre-training for table semantic parsing. In *International Conference on Learning Representations (ICLR)*, 2021.

Zhiyuan Zeng, Hamish Ivison, Yiping Wang, Lifan Yuan, Shuyue Stella Li, Zhuorui Ye, Siting Li, Jacqueline He, Runlong Zhou, Tong Chen, Chenyang Zhao, Yulia Tsvetkov, Simon Shaolei Du, Natasha Jaques, Hao Peng, Pang Wei Koh, and Hannaneh Hajishirzi. RLVE: scaling up reinforcement learning for language models with adaptive verifiable environments. In *International Conference on Machine Learning (ICML 2026)*, 2026.

Yunpeng Zhai, Shuchang Tao, Cheng Chen, Anni Zou, Ziqian Chen, Qingxu Fu, Shinji Mai, Li Yu, Jiaji Deng, Zouying Cao, Zhaoyang Liu, Bolin Ding, and Jingren Zhou. AgentEvolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*, 2025.

Guibin Zhang, Hejia Geng, Xiaohang Yu, Zhenfei Yin, Zaibin Zhang, Zelin Tan, Heng Zhou, Zhongzhi Li, Xiangyuan Xue, Yijiang Li, Yifan Zhou, Yang Chen, Chen Zhang, Yutao Fan, Zihu Wang, Songtao Huang, Francisco Piedrahita-Velez, Yue Liao, Hongru Wang, Mengyue Yang, Heng Ji, Jun Wang, Shuicheng Yan, Philip Torr, and Lei Bai. The landscape of agentic reinforcement learning for LLMs: A survey. *Transactions on Machine Learning Research*, 2026a.

Jipeng Zhang, Haolin Yang, Kehao Miao, Ruiyuan Zhang, Renjie Pi, Jiahui Gao, and Xiaofang Zhou. ExeSQL: Self-taught text-to-SQL models with execution-driven bootstrapping for SQL dialects. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 24305–24326, 2025.

Kai Zhang, Xiangchao Chen, Bo Liu, Tianci Xue, Zeyi Liao, Zhihan Liu, Xiyao Wang, Yuting Ning, Zhaorun Chen, Xiaohan Fu, Jian Xie, Yuxuan Sun, Boyu Gou, Qi Qi, Zihang Meng, Jianwei Yang, Ning Zhang, Xian Li, Ashish Shah, Dat Huynh, Hengduo Li, Zi Yang, Sara Cao, Lawrence Jang, Shuyan Zhou, Jiacheng Zhu, Huan Sun, Jason Weston, Yu Su, and Yifan Wu. Agent learning via early experience. In *Proceedings of the 43rd International Conference on Machine Learning (ICML)*, 2026b.

Meihui Zhang, Hazem Elmeleegy, Cecilia M. Procopiuc, and Divesh Srivastava. Reverse engineering complex join queries. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 809–820, 2013.

Yuge Zhang, Qiyang Jiang, Xingyu Han, Nan Chen, Yuqing Yang, and Kan Ren. Benchmarking data science agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. InfiniteWeb: scalable web environment synthesis for GUI agent training. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28465–28492, 2026c.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025a.

Fuheng Zhao, Jiayue Chen, Lawrence Lim, Ishtiyaque Ahmad, Divyakant Agrawal, and Amr El Abbadi. LLM-SQL-Solver: Can LLMs determine SQL equivalence? In *2025 IEEE International Conference on Big Data (BigData)*, pp. 1887–1894, 2025b.

Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. MultiHiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

Danna Zheng, Mirella Lapata, and Jeff Z. Pan. Archer: A human-labeled text-to-SQL dataset with arithmetic, commonsense and hypothetical reasoning. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2024)*, 2024a.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, and Ying Sheng. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37 (NeurIPS 2024)*, 2024b.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2SQL: Generating structured queries from natural language using reinforcement learning. *arXiv preprint arXiv:1709.00103*, 2017.

Yifei Zhou, Sergey Levine, Jason Weston, Xian Li, and Sainbayar Sukhbaatar. Self-challenging language model agents. In *Advances in Neural Information Processing Systems 38 (NeurIPS 2025)*, 2025.

Fengbin Zhu, Wenqiang Lei, Youcheng Huang, Chao Wang, Shuo Zhang, Jiancheng Lv, Fuli Feng, and Tat-Seng Chua. TAT-QA: A question answering benchmark on a hybrid of tabular and textual content in finance. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

Yuxuan Zhu, Tengjun Jin, Yoojin Choi, and Daniel Kang. Human-level text-to-SQL via reinforcement learning on verified data, without pipeline engineering. *arXiv preprint arXiv:2603.20004*, 2026.

Zilin Zhu, Chengxing Xie, Xin Lv, and slime Contributors. slime: An LLM post-training framework for RL scaling. `https://github.com/THUDM/slime`, 2025.

A TRAINING CONFIGURATION

We train with slime (Zhu et al., 2025), which runs Megatron-LM (Shoeybi et al., 2019) for optimisation and SGLang (Zheng et al., 2024b) for rollout generation. Both stages use GRPO (Shao et al., 2024) with full-parameter updates and the same optimisation settings: Adam at a learning rate of 1×10^{-6} with $\beta = (0.9, 0.98)$ and weight decay 0.01, a clipped surrogate with $\epsilon = 0.2$ and $\epsilon_{\text{high}} = 0.28$, truncated importance sampling clipped to $[0.5, 1.5]$, a low-variance KL loss against the initial policy with coefficient 0.01, and an entropy coefficient of 0.001. Rollouts are sampled at temperature 1.0 with a maximum of 8192 response tokens per action turn. Table A1 lists the settings in which the two stages differ.

Table A1: Settings that differ between the *EnvLunch* stage and downstream fine-tuning. $n_{\max}$ is the largest training budget of a benchmark.

	EnvLunch stage	Downstream fine-tuning
Tasks per step	16	8 labelled examples
Optimiser updates per step	4 (batch 32)	1 (batch 64)
Steps	100	$n_{\max}/8$
Turn limit	16 relational, 48 dataframe	per benchmark (Table A3)

The *EnvLunch* stage. The stage draws its tasks at random from the pool of Appendix C and consumes 1600 tasks in a single pass.

Downstream fine-tuning. A fine-tuning run takes up to 11 hours of training, depending on the number of labelled examples and the rollout length during training. During fine-tuning, both models see the exact same examples in the same order. We evaluate the checkpoints whose n is a power of two, up to the largest power of two that the pool of the benchmark holds. The two models on a curve differ only in their initial weights; seed, data order, limits, decoding and scorer are shared. Evaluation uses the official scorer on the full held-out set.

B TRAINING COST

EnvLunch adds a reinforcement learning stage before downstream fine-tuning. The warm-up runs 100 steps of 16 tasks and takes about 4 hours for the 9B model and 3 hours for the 4B model. Each warm-up step involves multi-turn rollouts and executes every submitted trajectory in K perturbed copies of the environment. This cost is paid only once for each backbone. All downstream benchmarks, at every training budget and with every random seed, start from the same warm-up checkpoint, so a single warm-up serves all thirteen benchmarks of Section 5. Table A2 compares this one-off cost with the extra fine-tuning the baseline needs to reach the accuracy that the *EnvLunch* model attains at its smallest training budget. Baseline n denotes the number of fine-tuning tasks required for the baseline to match the performance of *EnvLunch* at its smallest budget. Where the baseline never reaches the target, we use its largest budget, which gives a lower bound. Summed over the thirteen benchmarks, this extra fine-tuning takes about 7 hours at 9B and 4 hours at 4B, more than the warm-up itself.

C WARM-UP TASK SYNTHESIS

Environments. The relational corpus holds 112 databases: 63 from the Spider training split (Yu et al., 2018) and 49 from the BIRD training split (Li et al., 2023); the 7 Archer (Zheng et al., 2024a) training databases are Spider databases among the 63. The dataframe corpus holds 398 tables: 351 reinforcement-learning training tables released with DataMind (Qiao et al., 2026), 32 BIRD training databases read as dataframes, and 15 DABench tables (Hu et al., 2024a). The builder excludes by name the development databases of Spider and BIRD and every table in the DABench evaluation manifests, and the pool used for training shares no environment with any evaluation set.

Table A2: The one-off warm-up compared with the extra fine-tuning the baseline needs to reach the accuracy of the *EnvLunch* model at its smallest training budget on the same GPU node. n is the number of annotated examples the baseline needs to reach the *EnvLunch* performance at its smallest training budget.

	9B		4B	
Benchmark	Baseline n	Extra time	Baseline n	Extra time
ConvFinQA	512	22 min	128	3 min
TAT-QA	128	6 min	128	6 min
MultiHiertt	512	28 min	256	32 min
TableBench	128	8 min	256	9 min
DSEval	never	$>$ 6 min	never	$>$ 5 min
DataBench	256	11 min	512	22 min
BIRD	1024	123 min	1024	139 min
Spider	512	15 min	64	3 min
Spider2-lite	never	$>$ 8 min	8	0 min
HotpotQA	never	$>$ 65 min	512	20 min
ALFWorld	128	15 min	64	5 min
AppWorld	never	$>$ 19 min	8	0 min
SpreadsheetBench	never	$>$ 85 min	8	3 min
Total extra fine-tuning		$\approx$ 7 h		$\approx$ 4 h

Action pool. The operators are those of SQL and `pandas`, and nothing in the pool is specific to a database. The SQL side has fifteen operators. Ten of them aggregate, at increasing levels of composition: a grouped aggregate, then a filter before it, a foreign-key join, a threshold recomputed from that join and grouping, a scalar subquery as the threshold, a second grouping key, a conditional count beside a plain aggregate, a window ranking inside a subquery, a three-table foreign-key chain, and a common table expression compared against its own average. The other five do not aggregate: projection under a filter on a sampled literal, the rows where a column is null, a top-k ranking with a tie-break, grouping by the year of a date column, and projection across one foreign-key hop. The `pandas` side composes one source action (a single table, or two tables merged on a foreign key), a chain of transform actions (filters by a quantile threshold or a category value, top-k, grouped aggregates on one or two keys, a ratio of two columns, a rank column) and one terminal action (a scalar aggregate, a quantile, a distinct count, a row count, or a small mapping produced by a grouping). The difficulty level sets the chain length to 0, 1–2, 3–4 or 5–6 transforms, with a join added at the two highest levels with probability one half.

Online filtering. An action enters the candidate pool only if its operands exist in the current observation and it can still change it, and its parameters are read off that observation, so a threshold is an actual quantile of the column and a category filter names a value the column holds. A trajectory is kept only if it executes and yields a non-degenerate result in every perturbed copy, and its result changes in at least one copy, so that a probe that hardcodes the target output cannot obtain the full reward.

Perturbed copies and reward. A copy deletes 30% of the rows of every table with at least twenty rows, cascading along foreign keys so that the copy stays referentially consistent. On the dataframe side every measure column is also rescaled by a random factor; identifier and key columns are never changed. The agent never sees a copy. We use $K = 2$ copies on the relational side and $K = 3$ on the dataframe side. The agreement m of Equation 4 is an overlap of row multisets for relational results and an exact comparison after canonicalisation for dataframe results. The turn limit $t_{\max}$ is 16 for SQL and 48 for `pandas`. The brevity weight of Equation 4 is $w = 0.5$.

Without perturbed copies. In a preliminary run, we trained the warm-up model with the perturbed copies removed, so that the reward compares execution results on the original database only, and evaluated it on 208 held-out warm-up tasks from 12 unseen databases, scored in perturbed copies. Accuracy falls to 16.8, below 28.1 for the untrained model and 46.1 with the copies. All 163

answers to query tasks are constants with no `FROM` clause, against one with the copies, and 78% of episodes submit on the first turn, against none with the copies.

D BENCHMARKS

Table A3 gives the metric, held-out set, turn limit and training budgets n of every benchmark, shared by both scales and both models. The labelled pool is drawn from the training split and subsampled to at most 2048 examples, and the held-out set is the evaluation split the benchmark ships. DSEval, Spider2-lite and SpreadsheetBench have no usable training split; we split them at the level of the underlying data source (Spider2-lite by database), with disjoint identifiers on the two sides, which leaves pools of 99, 74 and 493 examples.

Table A3: Per-benchmark configuration. $t_{\max}$ is the turn limit and *Held-out* the size of the evaluation set.

Benchmark	Metric	Held-out	$t_{\max}$	n range
ConvFinQA	Proportional match	1486	16	16–1024
TAT-QA	Proportional match	494	12	16–1024
MultiHiertt	Proportional match	300	32	16–1024
TableBench	Proportional match	118	16	16–512
DSEval	Proportional match	87	20	8–64
DataBench	DataBench accuracy	351	16	16–1024
BIRD	Execution accuracy	1534	32	16–2048
Spider	Execution accuracy	1034	32	16–2048
Spider2-lite	Execution accuracy	61	8	8–64
HotpotQA	Proportional match	400	32	16–1024
ALFWorld	Success rate	134	30	16–1024
AppWorld	Task goal completion	57	30	8–64
SpreadsheetBench	Execution accuracy	394	6	8–256

D.1 BENCHMARKS BEYOND AGENTIC DATA ANALYSIS

On BIRD, Spider and the local SQLite subset of Spider2-lite, each turn either runs a read-only SQL query, whose rows or error message is returned, or submits a final query. HotpotQA uses the distractor setting, and the held-out set is 400 questions sampled from the validation split. ALFWorld takes text commands and returns the game feedback with the list of admissible commands, and is evaluated on the 134 unseen games. In AppWorld the agent writes Python that calls the application APIs, including those that return their documentation, and the episode ends when it calls the task-completion API. In SpreadsheetBench the agent writes `openpyxl` code that produces an output workbook, and it may inspect a recalculated preview of its own output before submitting. Figures A1 and A2 show the curves behind Table 2.

E THE SINGLE-EXAMPLE SWEEP

An example qualifies only if both models show a non-zero spread of rewards within its group, so that both are learning from it. Among the qualifying examples we take those at the 25th, 50th and 75th percentile of the per-group accuracy of the baseline. Table A4 gives the eighteen resulting curves; the $s = 0$ column precedes any update and is common to the three examples.

F ALTERNATIVE EXPLANATIONS

F.1 OTHER USES OF THE WARM-UP ENVIRONMENTS

Each model below replaces the *EnvLunch* stage and then enters the same downstream fine-tuning on the six data-analysis benchmarks, with the budgets of the main results (Table A5, Figure A3).

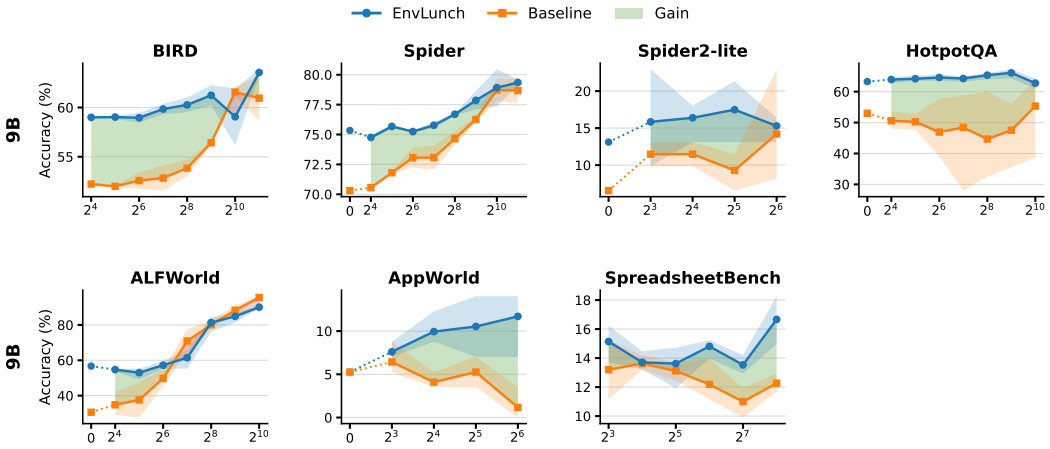

Figure A1: Fine-tuning curves at 9B for the seven benchmarks beyond agentic data analysis, with the axes and shading of Figure 2.

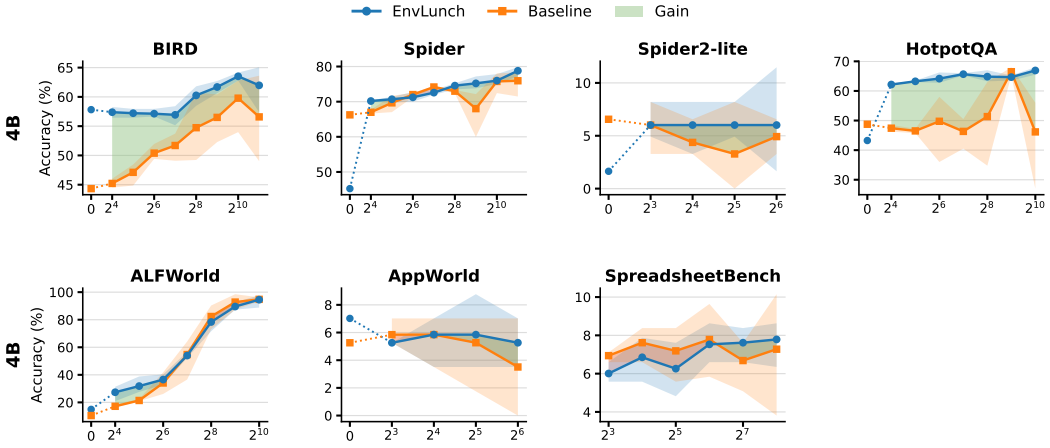

Figure A2: Fine-tuning curves at 4B for the seven benchmarks beyond agentic data analysis, with the axes and shading of Figure 2.

Continued pre-training. A template sampler opens each table of the dataframe corpus and writes a document with the file name, the columns and their types, a templated `pandas` expression and the result of executing it. The corpus holds 34,267 documents (6.60M tokens), and we train on it for one epoch of full-parameter next-token prediction.

Rejection sampling fine-tuning. For 1600 tasks of the *EnvLunch* pool, we sample up to 8 attempts for each task from the backbone under the *EnvLunch* limits and scoring, and keep those attempts with full reward. We fine-tune on them for one epoch, with the loss on model tokens only.

Synthesised tasks. The *EnvLunch* stage is kept unchanged except for the task pool: a backbone of the same scale generates the same number of tasks over the same environments. In *Self-Challenging*, the model observes the environment information, including field descriptions, table schemas or creation statements, and sample data, and directly generates questions and answers; tasks are dropped if the generated program fails. In *Explore-First*, the model first explores the environment freely, then generates questions and answers based on its interaction trajectory. In *Self-Play*, the model similarly

Table A4: One labelled example at three difficulties, 9B. Accuracy difference (*EnvLunch* minus baseline) after s optimisation steps on that example.

Benchmark	Example	$s=0$	$s=1$	$s=2$	$s=4$	$s=8$	$s=16$	$s=32$
ConvFinQA	25th percentile	+20.5	+20.2	+19.0	+12.1	+3.0	+0.7	+18.0
	50th percentile	+20.5	+20.7	+19.2	+15.2	+7.1	+1.9	+18.0
	75th percentile	+20.5	+21.3	+21.7	+20.1	+12.7	+11.5	+22.3
TAT-QA	25th percentile	+36.0	+37.2	+36.4	+28.1	+13.0	+2.2	-0.2
	50th percentile	+36.0	+35.6	+38.1	+32.6	+33.0	+38.0	+36.4
	75th percentile	+36.0	+37.6	+35.4	+19.2	+2.8	+1.2	-0.4
MultiHiertt	25th percentile	+17.0	+18.3	+17.7	+20.3	+20.3	+18.3	+11.7
	50th percentile	+17.0	+16.7	+14.7	+11.3	+0.7	-9.3	-6.3
	75th percentile	+17.0	+15.7	+17.3	+17.3	+10.0	-7.0	-2.0
TableBench	25th percentile	+3.4	+2.5	+11.0	+4.2	+2.5	+0.8	+9.3
	50th percentile	+3.4	+0.0	+7.6	+5.1	+11.9	-0.8	+5.1
	75th percentile	+3.4	+1.7	+5.9	+5.1	+4.2	+10.2	+7.6
DSEval	25th percentile	+10.0	+1.0	+6.6	+4.6	+1.5	+1.9	+3.1
	50th percentile	+10.0	+7.3	+4.8	+10.0	+10.0	+17.6	+6.1
	75th percentile	+10.0	+11.5	+2.7	+4.9	+4.8	+2.5	+0.6
DataBench	25th percentile	+4.0	+6.3	+5.7	+4.8	+2.0	+6.0	+10.8
	50th percentile	+4.0	+3.7	+4.0	+7.4	+10.5	+19.9	+16.5
	75th percentile	+4.0	+3.4	+2.9	+2.6	+5.1	+5.4	+4.6

observes the environment information and generates questions and answers, but additionally acts as the solver, attempting each task eight times and using majority voting across the attempts as the ground-truth answer; tasks are dropped when all eight attempts agree or fewer than three attempts agree. Following the respective designs, we construct all three task-synthesis pipelines over the same environments while keeping the harness, tools, and interaction limits identical to those of *EnvLunch*, so they differ from *EnvLunch* only in how tasks are constructed.

Table A5: Curve-level gain over the baseline for each stage placed before downstream fine-tuning, averaged over the six data-analysis benchmarks.

Stage before downstream fine-tuning	4B	9B
None (baseline)	0.0	0.0
Continued pre-training on environment text	−3.5	−2.4
Rejection sampling fine-tuning on successful trajectories	−5.7	−4.7
RL on Self-Challenging tasks	+5.2	+1.6
RL on Explore-First tasks	+5.3	−0.3
RL on Self-Play tasks	+6.4	−4.0
EnvLunch	**+9.6**	**+9.3**

G TRAJECTORY READING

We pair the trajectories of the two models on 78 items that both score identically before fine-tuning, and present each pair without scores, with the two models labelled *A* and *B* by a hash of the item name. The reader model (Claude Sonnet 5) describes in free text the behaviours in which the two trajectories differ, with no taxonomy supplied. We check these descriptions against the trajectories, group them into behaviour categories by hand, and count on the categories; Figure 4 reports the counts for each model.

H THE CONTROLLED ENVIRONMENT FOR MEASURING UNCERTAINTY

Section 6 asks how much each action tells the model about its environment. To measure this, we need tasks in which we know exactly what the model does not know when it starts. We describe such a task through one example; all tasks are built the same way.

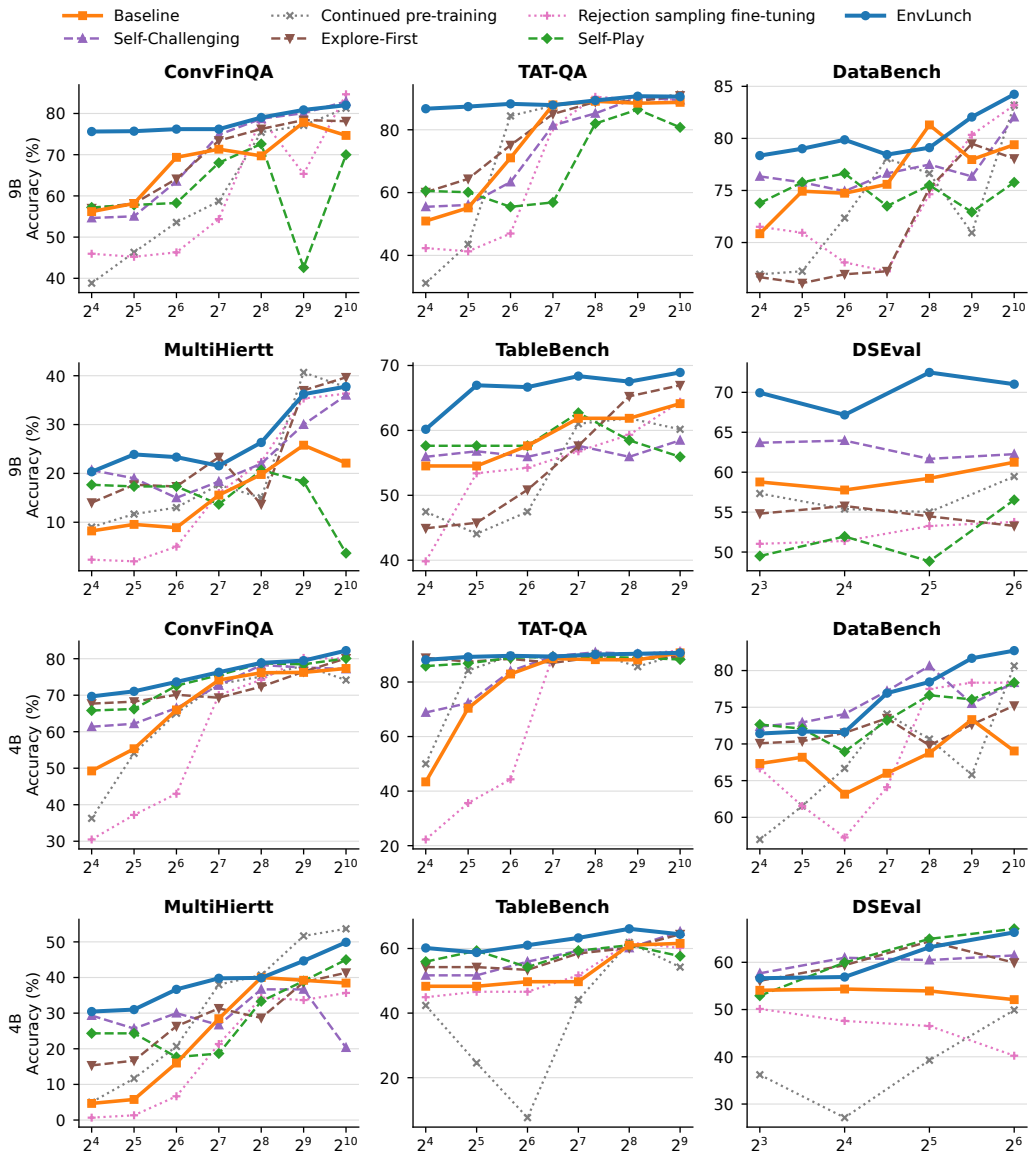

Figure A3: Downstream fine-tuning curves after each stage of Table A5, at 9B (top) and 4B (bottom).

The task. The model receives a sales table and the question "Consider the orders placed in Marlborough. Keep only those whose status is completed and whose product is Elm. What is their total amount?" The table has the columns order identifier, town, status, product, quantity and amount, and covers 20 towns with 12 orders each, except Marlborough, which has 60. As in DataBench, the prompt contains the column documentation and the first rows of the table, and the model runs code on the table before it reports a single number.

The difficulty. The documentation says that the town column contains names such as Marlborough, that the status is completed or cancelled, and that the product is one of five names including Elm. It adds that the records of some towns were imported from an older system and may use different codes in the town, status and product columns, a common data-quality problem (Rahm & Do, 2000; Kim et al., 2003), but it does not say which towns or which codes. In fact, the 60 Marlborough records write each of the three columns in one

of four ways (Table A6). In our example they might write the town as `MLB`, the status as `C` or `X`, and the product as `SKU-ELM`. The code `df[(df.town=="Marlborough") & (df.status=="completed") & (df.product=="Elm")]` then finds no orders, and the model has to find out how the three columns are written before it can compute the answer.

Table A6: The four ways of writing each column in the Marlborough records. The first is the one given in the documentation.

Column	Documented	Other ways
Town	Marlborough	MLB, Mrlbr, Mboro
Status	completed / cancelled	fulfilled / refunded, C / X, done / void
Product	Elm	SKU-ELM, E, Elm-W

Possibilities over parallel tables. Before its first action, the model has no way to tell which of the four ways each column uses. The preview rows never show Marlborough, the note in the documentation names no code, and the documentation lists two other towns with the same first letters (Marlow and Marlston), so selecting the towns whose name starts with "Marl" does not isolate Marlborough either. From the model's point of view, the table may therefore be any of $4 \times 4 \times 4 = 64$ versions, which differ only in how the three columns of the Marlborough records are written. All 64 have the same prompt and the same correct answer, since writing a value differently does not change which orders are selected. We generate all 64 versions and let the model work in one of them, never the one that writes all three columns as documented, so an answer computed from the documentation alone is always wrong. Every code in use appears in the Marlborough records, so the model can find out which version it is in by looking at them.

Measuring uncertainty. Because we hold all 64 versions, we can check which of them agree with what the model has seen so far. After each action, we run the model's code on every version and keep the versions on which it produces the same output as the model observed. We define the remaining uncertainty as $\log_2$ of the number of versions kept: 6 bits at the start, when all 64 are possible, and 0 bits when only the model's own version remains, that is, when it knows how all three columns are written. Continuing the example, where the model's version writes the town as `MLB`:

- The filter on the documented values above returns no orders in the model's version and in 62 others. Only the version that writes everything as documented would have returned orders, so this action rules out one version and the uncertainty falls from 6 to $\log_2 63 \approx 5.98$ bits.
- Listing the distinct values of the town column shows `MLB`. The 48 versions that write the town as Marlborough, `Mrlbr` or `Mboro` would have shown a different list and are ruled out. The 16 versions that remain differ only in the status and product columns, and the uncertainty falls to $\log_2 16 = 4$ bits.

Each column thus accounts for 2 of the 6 bits, and a model that inspects the data well identifies all three with a few actions.

Scale and evaluation. Six domains (sales, hospital visits, library loans, shipments, staff records and energy readings) change only the column names, town names and value ranges, and each contributes 25 tasks. The models in Figure 5 are evaluated right after their warm-up, without any downstream fine-tuning, in the DataBench harness with at most 24 turns. Each task is run once with greedy decoding and twice with sampling (temperature 0.7, top-p 0.8). The three runs of a task are averaged, and the 95% intervals in Figure 5 come from resampling tasks (bootstrap).

I LIMITATIONS

Generality across environments. *EnvLunch* is evaluated on relational databases and dataframes, and we leave a broader empirical study across environment types to future work. The underlying construction, however, is not specific to data analysis. It only requires an executable action space from which valid actions can be sampled and a way to evaluate the resulting observations. Such

action spaces are readily available in many agent environments. For example, web environments expose clickable elements, text inputs, scrolling, and navigation actions that can be checked against the current page state. Many application and tool-use environments similarly expose structured operations with well-defined execution outcomes. Moreover, counterfactual environments can often be constructed by modifying relevant state variables while keeping the environment otherwise unchanged. A number of general agent environments are also backed by structured databases or state stores, which can provide a natural substrate for such perturbations. Extending *EnvLunch* to these environments is therefore conceptually straightforward, although we leave systematic evaluation beyond databases and dataframes to future work.

Open-ended downstream tasks. Our warm-up formulation uses an executable outcome that can be automatically compared, which makes it particularly natural for tasks with structured or verifiable outputs. More open-ended downstream tasks, such as report writing, chart generation, or free-form question answering, may require a richer agreement function. One natural extension is to use rubric-based evaluation or other structured semantic criteria to compare outcomes. Investigating how environment-derived warm-up can support such open-ended tasks is an interesting direction for future work.

Model scale. Our experiments focus on Qwen3.5-4B and Qwen3.5-9B. These models provide a practical regime in which the effect of environment-driven warm-up can be studied across multiple tasks and training settings under our available computational budget. We deliberately emphasise breadth and robustness of evaluation at this scale, including six agentic data-analysis benchmarks, seven additional agentic tasks, multiple random seeds, controlled comparisons, and behavioural analyses. Whether the same gains persist across substantially larger models and other model families remains an important direction for future work.

Behavioural analysis. Our behavioural analysis combines automatic trajectory comparison with language-model-assisted description and manual verification. For the trajectory analysis, we compare only pairs on which the baseline and *EnvLunch* model obtain the same score, and randomly swap the order of the two trajectories before analysis to reduce potential presentation bias. The language model descriptions are then manually checked against the trajectories, and this verification identifies no erroneous description. The analysis focuses on the most prominent behavioural differences and may therefore omit rarer or more subtle long-tail behaviours. A more comprehensive taxonomy of interaction strategies and their relationship to downstream performance is left for future work.

J A WARM-UP ROLLOUT

This appendix shows one rollout sampled about three quarters of the way through the warm-up of the 9B model. The environment is the `ice_hockey_draft` database from the BIRD training split, which records draft prospects in `PlayerInfo` and their statistics for each season in `SeasonStatus`. The hidden trajectory joins the two tables, keeps the players who appear in the 1997–1998 season, and returns the first 20 of them by ID with their birth year. The agent sees only the database and these 20 rows. Figure A4 gives the rollout.

The agent treats the players in the target as a set to be characterised. After a filter on birth year admits players outside the target, it opens the record of one target player and compares target players with non-target players on one attribute at a time, first the CSS rank, then nationality and draft year. Every target player was drafted in 1998. Reading the target as the complete set of qualifying players, the agent counts 214 players drafted in 1998 against 20 target rows, concludes that the draft year is necessary but not sufficient, and continues through draft round, height, weight and rank. In turn 14 it opens the season records of a target player, finds the 1997–1998 season, and commits a query that keeps the players drafted in 1998 who played in that season.

On the original database the two conditions select the same 214 players, so a query on the draft year alone reproduces the target exactly. The perturbed copies separate them, because row deletion removes season records of players who remain in the player table. The query on the draft year alone then returns 17 and 19 of the 20 target rows in the two copies, while the committed query agrees with the hidden trajectory in both.

Target observation (20 rows of `ELITEID, birthyear`): `[90, 1979], [97, 1980], [110, 1974], [114, 1980], [137, 1977], ..., [1934, 1978]`

	Agent	Observation
1	`SELECT ELITEID, birthyear FROM PlayerInfo ORDER BY rowid LIMIT 20`	`[9, 1981], [18, 1980], [27, 1983], ...`
2–3	`SELECT ELITEID, birthyear FROM PlayerInfo WHERE birthyear LIKE '197%' OR birthyear LIKE '1980%' ORDER BY ELITEID`	`[18, 1980], [59, 1980], [71, 1972], ...`
4	*The results include IDs that are not in the target table. Let me inspect the specific IDs from the target table to find the filtering criteria that distinguishes them.* `SELECT * FROM PlayerInfo WHERE ELITEID = 90`	`[90, 'Matthias Trattnig', '1979-04-22', ..., 'Austria', ..., 1998, 4, ...]`
5–7	*Let me check the CSS_rank values for IDs in the target table and IDs not in it.* `SELECT ELITEID, CSS_rank FROM PlayerInfo WHERE ELITEID IN (9, 18, 27, ..., 74) ORDER BY ELITEID`	`[9, 192], [18, 176], [27, 24], ...`
8	*Let me look at draftyear or nation.* `SELECT ELITEID, nation, draftyear FROM PlayerInfo WHERE ELITEID IN (90, 97, ..., 1934) ORDER BY ELITEID`	`[90, 'Austria', 1998], [97, 'Sweden', 1998], [110, 'Sweden', 1998], ...`
9	`SELECT ELITEID, draftyear FROM PlayerInfo WHERE draftyear = 1998 ORDER BY ELITEID`	`[90, 1998], [97, 1998], [110, 1998], ...`
10	`SELECT COUNT(*) FROM PlayerInfo WHERE draftyear = 1998`	`[214]`
11–13	*There are 214 players with draftyear = 1998, but the target only has 20 IDs. So draftyear = 1998 is necessary but not sufficient.* `SELECT ELITEID, draftyear, draftround FROM PlayerInfo WHERE ELITEID IN (90, 97, ..., 1934)`, then height and weight, then CSS_rank	`[90, 1998, 4], [97, 1998, 8], [110, 1998, 3], ...`
14	*Let me look more systematically, maybe checking for players who appear from multiple seasons or specific criteria on the SeasonStatus table.* `SELECT * FROM SeasonStatus WHERE ELITEID = 90 LIMIT 1`	`[90, '1997-1998', 'Univ. of Maine', 'NCAA', ...]`
15	*Maybe the target IDs are those from the 1997-1998 season.* (a further query over the target IDs)	`[90], [97], [110], ...`
16	**Answer:** `SELECT ELITEID, birthyear FROM PlayerInfo WHERE ELITEID IN (SELECT ELITEID FROM SeasonStatus WHERE SEASON = '1997-1998') AND draftyear = 1998 ORDER BY ELITEID LIMIT 20`	*agrees with the hidden trajectory in every perturbed copy*

Figure A4: A rollout of the 9B model on the warm-up task of Appendix J. Italic text is the agent's reasoning, lightly shortened; queries are verbatim except for elided ID lists, and observations show their first rows.