EFFICIENT TEST-TIME ADAPTATION THROUGH HUMAN-AGENT INTERACTION

Anonymous authors
Paper under double-blind review

ABSTRACT

AI agents are trained on population-scale data to encode broad capabilities spanning those of many practitioners. Yet the artifacts they produce rarely meet the personal bar professionals need to stake their reputation on. On realistic, open-ended tasks where success criteria are heterogeneous and insufficiently documented, individual expertise lives precisely in the elevation and departure from the average. In practice, iterative human-agent interaction surfaces criteria that users cannot fully specify up front, yet apply repeatedly across tasks. We argue this cross-session interaction data is a rich, underused signal for closing the gap to individual expertise. In this work, we propose test-time adaptation through human-agent interaction (TAHI), which integrates these signals into agent context and weights, and crystallizes each user's training and evaluation criteria via an evolving rubric module. We adapt agents to 30 individuals in two high-utility domains, writing and visual creation, on a total of 600 tasks. Our agents improve on solo task success by 4.5–20.9% within only tens of tasks. Meanwhile, our evolving rubric module serves as a scalable annotation tool, creating evaluation rubrics that catch 16.0–22.3% more failures than those from LMs or humans alone. While agents are adapted towards individuals, we show these personalized agents also produce improvements in success of up to 8.8% that generalize across users.[1]

1 INTRODUCTION

Large language model (LLM) based agents have rapidly matured into capable generalists, producing reasonable outputs across writing, software engineering (Wang et al., 2025d), and a growing range of occupational tasks (Patwardhan et al., 2026). Yet this general competence still falls short of expert standards in notable ways: flattening responses into homogeneity (Jiang et al., 2026) or drifting away from users' original opinions (Abdulhai et al., 2026), yielding outcomes that rarely meet the bar for professionals to use without further iterative editing.

Human expertise, particularly the tacit practice and judgment accumulated over years of domain experience, is only partially captured by the population-scale textual corpora used to train foundational LMs (Gobet, 1998). A model necessarily learns broad capabilities spanning many practitioners, but an individual professional needs specific knowledge, judgment, and strategies that elevate and depart from the population average. This tension only sharpens on realistic, open-ended tasks, which have diverse success criteria that are seldom fully documented (Weidinger et al., 2025), leaving a constant gap between the capabilities an agent can learn from documented human knowledge and those that remain unverbalized. Compounding this, expertise is inherently individual and can conflict across practitioners, yielding a set of desiderata that is hard for a single trained model or default agent framework to satisfy simultaneously. Closing this last mile, then, requires agents to adapt at deployment time, to the individuals they are working with (Kojima et al., 2021; Hawkins et al., 2020). Moreover, it is important to do so efficiently, converging on a user's preferred way of working within a handful of sessions, before their patience for dozens of trial rounds runs out (Shaikh et al., 2025).

Efficient test-time adaptation, however, demands the right learning signal. When agents improve solely by reflecting on their past attempts, progress plateaus in the absence of new signals from the outside (Huang et al., 2024; Xu et al., 2024b). It often takes many rounds of human-agent interaction

[1] https://anonymous.4open.science/status/tahi-agent-adaptation-DCF2

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

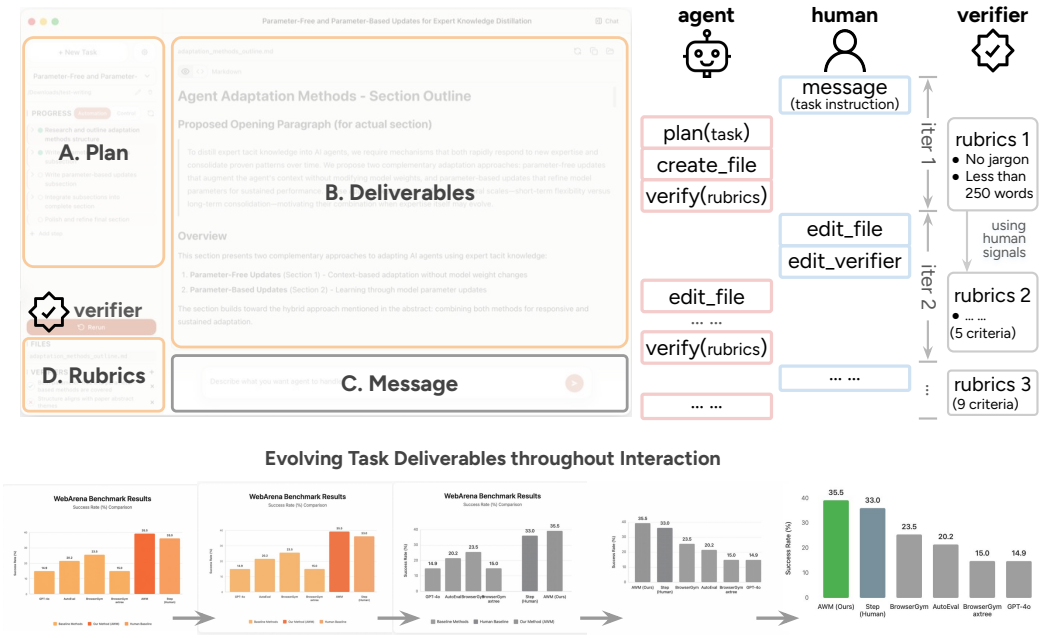


Figure 1: Illustration of a human-agent interaction process using our interface with an evolving verifier module. The agent executes tasks, while the human provides instructions, suggestions, and corrections via four modules: plan adjustment (A. Plan), output file editing (B. Deliverables), textual feedback (C. Message), and criteria specification (D. Rubrics). Human and agent iterate on this instruction-execution loop to improve the deliverables (iter 1 $\rightarrow$ iter 2 $\rightarrow$ ...) until the human is satisfied with it (bottom). Meanwhile, the verifier module evolves by integrating human signals from the previous iteration, updating the rubrics to capture human preferences.

to iterate toward a target shape the user could not fully specify up front. Many of these signals recur across task sessions (Wang et al., 2025c), yet current agent systems largely leave them unused when interacting with humans. It is precisely this multi-session interaction data, accumulated through the process of working with the agent, that offers a rich, under-explored learning signal for agents. Unlike a one-off correction, we envision a human mentor working alongside a trainee agent across multiple task sessions, offering continual, hands-on suggestions across multiple channels. The challenge, then, is to collect and integrate these interaction streams into the agent.

To this end, we propose test-time adaptation through human-agent interaction (TAHI). Human interaction signals are processed in a streaming manner, where agents continually adapt themselves based on previous interaction sessions to solve subsequent ones, progressively reducing the interactions needed to meet the human experts' expectations (Figure 2). Our agent is supported by a trainable backbone LM (Qwen3.6-35B) conditioned on editable contexts, and an interface that affords various channels of human interaction (Figure 1). Notably, besides using human textual feedback (Song et al., 2026a) as in existing frameworks (our *Message* module), our interface enables versatile additional signals, including planning adjustments (*Plan*), verification specifications (*Rubrics*), and direct edits to produce artifacts (*Deliverables*); allowing us to study the value of versatile human signals.

Alongside the collection of human-agent interaction data, TAHI supports test-time agent adaptation and rubric creation for the open-ended tasks we evaluate on. For one, our agent allows (i) context adaptation, by inducing factual memory and procedural skills, that enables easy inspection and revision of agent learning by the user; and (ii) weight adaptation, by encouraging agents to produce human-guided final output over its initial one using direct preference optimization (DPO, (Rafailov et al., 2023)) on LoRA adapters (Hu et al., 2022), to capture more behavioral changes and spares the context from overgrowing. For another, we embed an evolving verifier module that uses an LLM to crystallize human activity into concrete evaluation criteria, while allowing users to correct anything that appears inaccurate. It serves as a scalable tool to annotate criteria that agents can be trained and evaluated against, and spares users the effort of hand-writing criteria for open-ended tasks (§2).

We individually adapt agents to 30 human experts across 600 tasks in two domains with wide AI use, namely writing and visual creation. For each domain, we collect 5 humans' interaction data

with our agent in three scenarios: online context adaptation, online weight adaptation, and offline (no adaptation); with each human doing 20 tasks in total. From this data, we find:

• **Agents effectively and efficiently adapt to individual users.** Under both context and weight adaptation, our agents achieve similarly substantial improvements of 4.5–12.9% and 4.5–20.9% in solo task success rate, and do so efficiently within only 20 task sessions. On additional held-out tasks, our context- and weight-adaptive agents improve by 3.2–6.6% and 4.8–5.2% on rubrics tied to the same humans, signaling strong cross-task generalization capabilities (§3).

• **Our evolved rubrics are more comprehensive than those created by LLMs or humans alone.** LLM-produced rubrics often miss critical evaluation aspects, and humans require substantial motivation and training to avoid the same pitfalls. Our evolved rubrics capture 16.0–22.3% more failures on imperfect agent solutions, compared to rubrics produced by LLMs and even humans alone (§4).

Looking forward, we envision a future where AI agents are not generic, but adaptive tools shaped by, and accountable to, the individuals they serve.

2 Efficient Test-Time Adaptation through Human-AI Interaction

2.1 Fundamentals: Human-Agent Interaction

We formalize the problem as a human-agent interaction (Cukurova, 2026) where, given a task specified by the human, the agent takes actions to execute it, while the human provides feedback and corrections along the way.

Modeling Agent and Human We define two parties involved in a task session. First, an LM-based agent A operating with policy $\pi_\theta^A(\cdot|c)$ parameterized by an LM θ and conditioned on context c. Following two threads of work adapting agents in context, we design two components: (i) memory storing factual and preferential knowledge (Xu et al., 2026), and (ii) skills storing procedural knowledge (Wang et al., 2024; Anthropic, 2025). As our goal is for agents to adapt towards the human, we also explicitly define human H operating with policy π^H.

Given a task with instruction q, both parties can take actions from their respective action space to form an action trajectory $\tau = [(s_0, o_0, a_0), (s_1, o_1, a_1), \cdots]$ that solves the task. s_i represents the shared environment state, o_i is the observation of the action-taking party (agent A or human H), and a_i is the action being taken.

For agents, we adopt the common action space for both GUI and coding activities. We adopt coding agents as generalist agents, taking the position that programmatic actions subsume common computer-use actions (e.g., `click`, `scroll`, `type`) and suffice for solving varied tasks.

$$\mathcal{A}^A = \{\texttt{message}, \texttt{file_create}, \texttt{file_edit}, \texttt{file_read}, \texttt{execute}, \texttt{plan}\}$$

This action space endows the agent with the following capabilities. (1) communication (`message`): surfacing information or requesting clarification via messages; (2) file manipulation (`file_create`, `file_edit`, `file_read`): reading, creating, or editing files in the environment; (3) program execution (`execute`): executing program scripts; and (4) planning (`plan`): creating and refining structured step decomposition, as well as verification criteria for each step.

The human acts within a complementary action space:

$$\mathcal{A}^H = \{\texttt{message}, \texttt{file_edit}, \texttt{plan_edit}, \texttt{context_edit}, \texttt{verify}, \texttt{trigger}\}$$

Humans share a few actions similar to agents: (1) communication (`message`): providing feedback or information to the agent via text messages; (2) file manipulation (`file_edit`): editing agent-produced artifacts; (3) planning (`plan_edit`): modifying the agent-proposed step decomposition and the verification criteria; Meanwhile, humans are equipped with more actions for supervising and correcting the agent, via: (4) verification (`verify`): providing pass/fail signals to criteria; (5) edit agent context (`context_edit`): edit the content in memory and skills files; and (6) agent solution refinement (`trigger`): signals the end of human activities and triggers the agent to act further.

The entire task session τ can be viewed as interleaved $J + 1$ turns conducted by agent and human, where each turn is itself a trajectory of actions, τ_*^*'s, composed of one or more actions. Formally,

$\tau = [\tau_0^H, \tau_0^A, \tau_1^H, \tau_1^A, \cdots, \tau_J^H, \tau_J^A]$, where $\tau_0^H = [\texttt{message}(q)]$ represents the user task instruction step, with q being the text of instruction. We determine the end of a task session if the user does not follow up or take further actions. If τ_J^H triggers another round of agent actions (i.e., last action in τ_J^H is `trigger`), τ_J^A will be non-empty; otherwise $\tau_J^A = \emptyset$, indicating the user is satisfied with the solution after their own actions.

Agent Interface Modification to Encourage Versatile Human Signals To make it easy for human users to express versatile signals while preserving a smooth user experience, we build on top of the widely adopted open-source Agent Cowork[2] and add three components:

First, *initial task planning* (Figure 1, Plan). We ask the agent to propose an initial task plan, displayed in the interface Plan module, so that human users can give targeted feedback on it. Concretely, this planning action decomposes the instruction into multiple steps, each represented as a textual description w^i, and comes with an output artifact f^i. For each step, we design an LM-supported VERIFIER module that initiates a list of NL verification criteria $V^i = \{v_k^i\}$ to judge how much f^i satisfies w^i. We allow users to edit the steps, files, and verifiers by adjusting the *Plan*, *Files*, and *Rubrics* modules in the interface. Formally, this action can be denoted as $a_{0,0}^A = \texttt{plan}(o_{0,0}^A) \rightarrow \{(w^i, f^i, V^i)\}$, where $a_{0,0}^A$ denotes the first action taken in the first agent turn of trajectory $\tau_0^A = [(s_{0,0}^A, o_{0,0}^A, a_{0,0}^A), \cdots]$. Since users mostly interact with the files and verifiers produced in the last step, for simplicity, we use f and V to denote the final artifacts and rubrics.

Second, *direct file editing* (Figure 1, Deliverables). Existing interfaces mostly require human feedback to be verbalized via messages (C. Message), which is limiting when a desired change is hard for humans to verbalize accurately. We switch the main panel to a display of agent-produced deliverables (e.g., written documents, data visualizations), allowing users to directly edit them. Since humans favor UI-based interactions despite agents often operating in programmatic ways (Wang et al., 2025d), for visual deliverables (e.g., figures in HTML), our interface allows users to drag elements directly via the UI, which are translated into code diffs for the agent's observation (Nilsson et al., 2026).

Third, *automatic step verification* (Figure 1, Rubrics). The agent fulfills step goal w^i by generating a trajectory τ^i to produce artifact f^i. The VERIFIER module grades f^i against the criteria V^i, producing a list of binary pass/fail scores as $\texttt{verify}(f^i, v_k^i) \rightarrow r_k^i \in \{0, 1\}$. The grading outputs will be rendered on the interface as red crosses or green checks for the user's information. Users can change the grading if they disagree with the automatic verification result by executing the `verify` action. It serves as an assistive tool for users to check intermediate task success, and lays the foundation for constructing the task evaluation rubrics.

The Iterative Nature of Task Sessions Particularly for artifact-driven tasks, the human-agent interleaving action-taking often happens for multiple rounds, to produce the final artifacts that satisfy the users. In response to this iterative nature of task sessions, we can view each pair of human-agent trajectories (τ_j^H, τ_j^A) as an interaction unit u_j, producing the output artifact f_j on top of the previous j iterations producing $f_0, \cdots, f_{j-1}$. Concretely, after each interaction unit u_j, the user can start another iteration by taking a sequence of actions forming τ_{j+1}^H. Once the user finishes acting and triggers, the agent will take in all past interactions $u_{0:j}$ and the latest human activity τ_{j+1}^H, to take actions τ_{j+1}^A and refine its own artifact output to f_{j+1}.

While the automatic verification module produces a quality measure r_0 of the current artifact, the initial rubrics V_0 are created in one pass by an LM and may not capture all criteria for faithful evaluation. Therefore, we enable evolution of the VERIFIER module to update the last-iteration criteria V_j using human activities τ_j^H, by verbalizing users' underlying intent and preferences into gradable rubrics, yielding updated rubrics V_{j+1}. In an ideal case, the final evolved verifiers V_J should capture all user requirements and serve as comprehensive evaluation rubrics, the quality of the resulting artifact f_j should be roughly increasing throughout task iterations (i.e., as j increases) by integrating human feedback, therefore $\sum_{v \in V_J} \texttt{verify}(f_{j'}, v) \geq \sum_{v \in V_J} \texttt{verify}(f_j, v), \forall j' > j$. Nonetheless, in practice, we observe this to hold when j and j' are farther apart, e.g., when $j = 0, j' = J$, which serves as a stronger agent learning signal.

[2] `https://github.com/DevAgentForge/Open-Claude-Cowork`

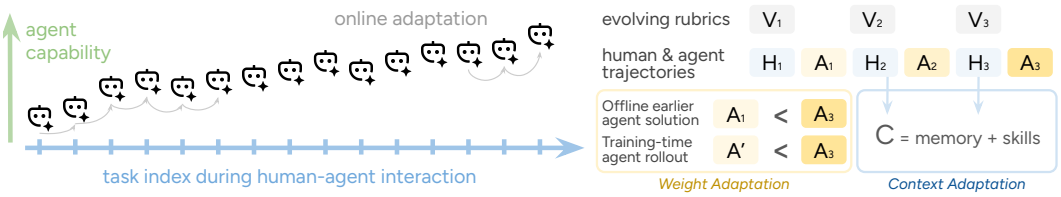


Figure 2: Left: test-time agent adaptation paradigm, illustrated with an example agent capability measure. Right: Implementation overview of context- and weight-based adaptation.

2.2 Test-Time Agent Adaptation Paradigm

Test-time agent adaptation operates in a streaming manner, where tasks arrive sequentially and each completed session provides new data for adapting the agent. Unlike one-time adaptation from a fixed dataset, the agent continually updates as it accumulates experience with the human.

Formally speaking, the adaptive agent processes a total of T tasks one by one: $t = 1, \cdots, T$. The agent is instantiated with an unadapted model and empty context, denoted as A^0, and gets updated after every task. Specifically, after the human and agent A^t co-finish the t-th task session, the agent A^t uses this session data to upgrade into A^{t+1} (§2.3). Next, human interacts with the latest agent A^{t+1} for the $t+1$-th task. As illustrated in Figure 2, we expect the agent's capability to roughly improve throughout the process of solving the stream of tasks.

To evaluate test-time adaptive agent solo success on a given task t, we adopt the first final-step artifact f_0^t produced solely by the agent in the first (index 0) task iteration u_0^t, and evaluate it against the final rubrics V_j^t to report the average success rate of all verifier scores. Notably, the agent producing this first artifact has only received supervision signals from previous tasks, but is exposed to nothing but the task instruction about the current task.

As an ablated setup for this *online* test-time agent adaptation, we also collect data from users interacting with non-adapting agents on the same K tasks, denoted as the *offline* data. We use this data to perform clean ablation studies in later experiment sections.

2.3 Agent Adaptation Methods

Given the agent policy $\pi_\theta^A(a|o, c)$, two adaptable components inside are the agent context c and backbone LM weights θ, which we denote as context- and weight-based adaptations.

Context-Based Adaptation transforms all human activities $\{\tau_j^H\}$ into verbalized textual information in two categories: (1) agent memory M storing factual and preferential knowledge, and (2) agent skill library K with procedural knowledge. At the t-th context adaptation, both inductions are powered by an LM conditioned on memory- and skill-specific instructions:

$$\texttt{induce}_{LM_{memory}}(\{\tau_j^H\}, M^t) \to M^{t+1}, \qquad \texttt{induce}_{LM_{skill}}(\{\tau_j^H\}, K^t) \to K^{kt1} \tag{1}$$

yielding an adapted policy $\pi_\theta^A(a|o, [M^t; K^t]) \to \pi_\theta^A(a|o, [M^{t+1}; K^{t+1}])$. Memory M captures declarative, user- or domain-specific information such as "the figure title should be bold-typed" or "the user prefers simple wording." Skill library K encodes multi-step workflows into reusable procedure knowledge, e.g., "to write a paper abstract: 1. write a motivating sentence, 2. introduce our method, 3. state numeric evidence, ...". See §G for our induction prompts.

Weight-Based Adaptation integrates human activities into model parameters θ. Unlike context-based adaptation that is bounded by what can be articulated as text and the model's ability to condition on it, weight-based approaches can potentially internalize extra patterns that are implicit and hard to verbalize, and directly train the model to produce different outputs.

Through preliminary exploration with various training algorithms (§C), we adopt Direct Preference Optimization (DPO) (Rafailov et al., 2023) and construct preference pairs by contrasting trajectories in the first and the last iterations τ_0 and $\tau_{0:J}$, where $\tau_{0:j}$ means the concatenation of trajectories $[\tau_0, \tau_1, \cdots, \tau_j]$. To prevent agents from relying on human interaction to achieve the correct task

solution, we consolidate $\tau_{0:j}$ into a shorter one-shot solution τ_j^{A*}, by merging multiple intermediate file edits into a single edit action with multiple arguments, and skipping the observational (e.g., read) and trigger actions. In this way, we are training the agent to prefer generating the final high-quality solution in one pass, without conditioning on intermediate human actions. Achieving this would mean agents are capable of one-shot user-preferred solutions by maintaining proper user modeling, and do not rely on human supervision further. Concretely,

$$\mathcal{L}_{DPO}(\theta) = -\log\sigma\left(\beta\left(\log\frac{\pi_\theta(\tau_J^{A*}|q_j)}{\pi_{ref}(\tau_J^{A*}|q_j)} - \log\frac{\pi_\theta(\tau_j^{A*}|q_j)}{\pi_{ref}(\tau_j^{A*}|q_j}\right)\right), j = 0 \tag{2}$$

where

$$\log\pi_\theta(\tau|q) = \sum_l log\pi_\theta(a_l|q, a_{<l}, s_{<l}; c) \tag{3}$$

a_l represents the action taken by the agent at step l, and s_l represents the resulting environment state. q_j is user query up to iteration j, in our setup, this is the initial task instruction when $j = 0$. σ is the sigmoid function, β is a hyper-parameter controlling how far π_θ can drift from π_{ref}. In implementation, we set the reference policy π_{ref} to be the initial policy before training on the task session. Although the policy does not explicitly condition on human activities, we do use the human activities to construct the supervision trajectory τ_j^{A*}. From each task session, we use the pair of the first and the last solutions as they perform the best empirically.

Training on these constructed pairs provides a limited number of training data points, we therefore augment the training pool with pairs constructed by sampling agent solutions during training time. Particularly, we treat the agent solutions sampled at training time as the rejected trajectory, and pair it with the final solution in the corresponding collected task session as the chosen trajectory, to create additional pairs for DPO training. See §C for more detailed ablations on DPO pairing.

3 Experiment: Adapting Agents to Human Expertise

In this section, we first introduce the experiment setup for test-time agent adaptation (§3.1), demonstrate its effectiveness using both the context and weight update methods with versatile human interaction signals, and further compare context and weight adaptation (§3.2).

3.1 Tasks, Setup, and Evaluation

Tasks. We construct tasks for two scenarios where AI agents have been relatively widely adopted, namely (i) paper abstract writing, given the title and introduction section of the paper, and (ii) data visualization: given an instruction of data and intended visualization style, create an HTML figure to realize the visuals.[3] For each scenario, we manually constructed 20 tasks using the best and outstanding papers of NeurIPS, ICML, ICLR, *CL, EMNLP, and CHI conferences in 2025 to ensure content quality and diverse topic coverage. Besides reporting test-time agent solo success rate on these 20 tasks, we further test how well agents can generalize to unseen tasks. We similarly create another 30 held-out tasks for both writing and data visualization scenarios, with the best papers from the same conferences in 2023 and 2024, covering different topics from test-time tasks (§B).

Setup For each task scenario, i.e., writing or data visualization, we recruit 5 human participants to interact with context-adaptive agents, and another 5 humans for weight-adaptive agents. To perform ablation studies, we recruit another 5 humans for non-adaptive agents, i.e., the offline setting. Each human interacts with the agent while completing 20 tasks within their assigned scenario. In summary, we collect 2 scenarios $\times$ 15 users $\times$ 20 tasks $=$ 600 data points.

For paper abstract writing, we recruit CS PhD students in five universities who are second-year or above and have experience in academic writing on topics spanning ML, NLP, and HCI. For data visualization, we recruit from the Handshake talent network,[4] where each participant has substantial experience in data analysis from professional backgrounds in healthcare, business analysis, engineering, or research in math, psychology, or biology. We initialize the agent with an empty context

[3] We are interested in domains (i) where certain amount of humans have greater, individualized expertise than an average AI agent, and (ii) are more open-ended to reflect the complex nature of real-world human activities.

[4] https://joinhandshake.com/

Scenario	Category	Test-Time Tasks			Held-Out Tasks	
		Baseline	Adaptive	Oracle	Baseline	Adaptive
Context	Abstract Writing	81.7	85.4*	89.5***	90.2	93.5***
	Data Visualization	77.0	86.9***	95.0***	89.9	92.1*
Weight	Abstract Writing	82.8	86.5*	91.1***	88.0	93.0***
	Data Visualization	69.0	81.1***	93.8***	86.2	91.2**

Table 1: Comparing agent solo success rates (SR) on paper abstract writing tasks and data visualization tasks, with test-time online adaptation via context (top) and weight (bottom). *Test-Time Tasks* report agents' performance on the 20 tasks we collected data on; *Held-Out Tasks* report agents' performance on 30 held-out tasks of the same categories. We performed significance testing on *adaptive* and *oracle* methods against the *baseline*, specifically two-sided paired t-tests. Notably, all adaptation results are statistically significantly better than the baseline, i.e., |t-statistic| > 2.0. We use *, **, *** to indicate results with $p < 0.05$, $p < 0.01$, and $p < 0.001$.

$c = \emptyset$ on top of the `Qwen3.6-35B-A3B` backbone LM hosted and trained with the Tinker API (tin, 2025). The context induction and evolving verifier are supported by `claude-sonnet-4-6`. [5]

Evaluation We test each test-time adapted agent in two folds. On the 20 tasks we collect interaction data on, we evaluate agent initial solutions f_0^k for each task k on its final evolved rubrics V_J^k, and calculate the average rubric scores throughout all tasks, to demonstrate its online adaptation ability. Note that our agent is adapted toward individual users e. We report the average scores among all users in the context and weight groups, respectively. To test if agent improvement generalizes across tasks, we also run the final adapted agent A^K on the 30 held-out tasks without any human involvement or further adaptation, and similarly take its produced artifacts for evaluation. To evaluate how well an agent adapts toward its corresponding user e, we need to construct dedicated evaluation rubrics V_e for these held-out tasks, as they are not similarly produced as the 20 test-time tasks we collect data on. Concretely, we ask the `claude-sonnet-4-6` LM to create rubrics for each task given its instruction, and ask it to cover (i) human-written rubrics: we ask user e to write rubrics generally applicable to all tasks in their tasking scenario (i.e., abstract writing, data visualization),[6] and (ii) summarized rubrics from the 20 tasks we collected from this user $\{V_e^k\}_{k=1}^{20}$. For summarized rubrics (ii), we use `claude-sonnet-4-6` to extract the important criteria shared across 20 tasks we collected data from. We ensure the generated held-out rubrics align with both the human-written and summarized rubrics, to ensure it satisfied this particular user's requirements.

In comparison, we run the *baseline*, namely an unadapted agent A^0 solo on the same test-time and held-out tasks independently, and evaluate the produced artifacts with the same rubrics. To get a sense of the upper-bound task performance the agent could possibly adapt to given this session data, we also take the last version of the artifact f_J as the *oracle* outcome from human-agent interaction. We similarly grade it with final rubrics V_J and report the task average scores. Empirically, as we later show in Table 1, this oracle score does not reach 100% often, due to limitations in the agent capability or allowed human effort to realize all human requirements.

3.2 Agents Can Efficiently Adapt to Human Expertise via Interaction

As shown in Table 1, both adaptation approaches substantially improve agent solo task-solving performance within just 20 task sessions. On the tasks we collect human-agent interaction data on (test-time tasks), our context- and weight-adaptive agents — evaluated on each task before the agent ever received human suggestions on it, using only adaptation accumulated from prior tasks — improve their solo task-solving success rate relatively by 4.5% and 4.5% on writing tasks, and 12.9% and 20.9% on data visualization tasks , respectively. To demonstrate effective improvements on this relatively small pool of 100 data points per task under each adaptation scenario, we conducted paired t-tests between baseline and adapted agents on matched task instances to assess the statistical significance of these gains. Notably, every paired test reaches significance ($t > 2.0$), demonstrating that the adaptation is substantially effective even from a small number of examples.

[5] We also tried contemporaneous open-source models in preliminary experiments, but they failed to produce reusable memory, skill, and verification criteria with desired quality.

[6] Due to resource limitations to ask each of the 30 humans to annotate rubrics for all 30 tasks.

To further test the cross-task generalization abilities of our adapted agents, we evaluate baseline and adapted agents on held-out tasks in Table 1 (right). Context- and weight-adaptive agents improve significantly over the baseline agent by 3.6% and 5.7% on writing and 2.4% and 5.8% on data visualization, relatively. These consistent improvements across task sets confirm that the adaptation generalized beyond the specific tasks during adaptation. Comparing the two task domains, we observe larger gains in data visualization than in writing, plausibly because the backbone LM's writing ability is already more extensively developed through pretraining, leaving less headroom for adaptation to improve upon. When transferring to held-out tasks within the same domain, writing gains remain comparable to its test-time performance, whereas gains on unseen data visualization tasks shrink noticeably. This asymmetry suggests that data visualization tasks are heterogeneous, and exposure to a wider variety of tasks will likely facilitate adaptation more.

Benefiting from Human Interaction Signals Beyond Feedback Much prior work reduced human feedback to textual messages sent back to the LM or agent (Christiano et al., 2017). Yet humans naturally express feedback through a much richer repertoire of activity, most of which remain unexplored as a source of agent learning signal. Through our interface, beyond text `message` actions, experts can directly edit files, adjust the agent's plan, revise verification criteria, and leave inline comments; each forming a distinct channel through which expertise surfaces. As shown in Figure 3, humans draw on this full repertoire throughout their interaction with agents.

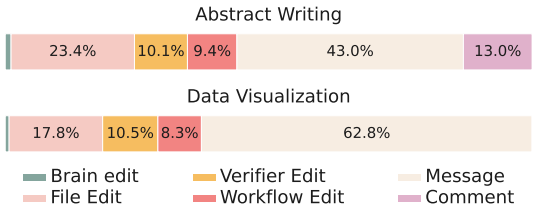


Figure 3: Distribution of human actions throughout their interactions with the agents. `comment` was not supported for data visualization tasks.

To test whether these diverse human actions serve as effective adaptation signals, we compare agent adaptation results using (i) all human actions and (ii) message actions alone. We run this comparison on collected offline data using context adaptation, since it is difficult to isolate the effect of non-message actions in the online scenario, or under the weight adaptation approach. (i) uses our default method for context adaptation, while (ii) restricts context induction input to message actions only. As shown in Table 2, adding non-message actions yields 3.0% and 7.6% more gains in writing and data visualization tasks, showing the value of actions beyond text feedback.

Method	Writing	DataViz
Baseline	80.8	66.4
All Actions	84.9	75.6
Msg Actions	81.9	68.0

Table 2: Comparing agent adaptation with all human actions and message actions alone.

Weight Adaptation Has Greater Inference-Time Efficiency than Context Adaptation Beyond task performance, weight adaptation offers a significant efficiency advantage at inference time. On the input side, weight-adaptive agents require 62.6% and 88.3% fewer tokens than their context-adaptive counterpart on writing and data visualization tasks, showing a direct benefit of encoding expertise in model weights rather than an ever-growing context.

This efficiency gain extends to the output side as well. Relative to task sessions from humans and unadapted baseline agents, weight-adaptive agents produce 9.1% and 4.4% fewer output tokens on writing and data visualization tasks, all while achieving better results. In contrast, context-adaptive agents move in the opposite direction, producing responses 20.1% and 6.1% longer on the same tasks, suggesting that context adaptation inherits some of the verbosity of the contexts it conditions on. In Figure 5, we investigate the deeper reason by analyzing the action content length by different action types, and find context-adaptive agents produce 143–176% and 69.9–95.7% longer `messages`, while rarely differ in other task execution actions, compared to the baseline and weight-adaptive agent.

4 Eliciting Human Expertise with Scalable Rubric Evolvement

Tasks beyond mathematical and programming problems have long been treated as "non-verifiable," owing to the difficulty of annotating a rubric that is both comprehensive and accurate enough for evaluation. How to properly evaluate such open-ended tasks remains a long-standing problem: recent work has largely converged on LLM-as-a-judge (Zheng et al., 2023) scored against rubrics, produced

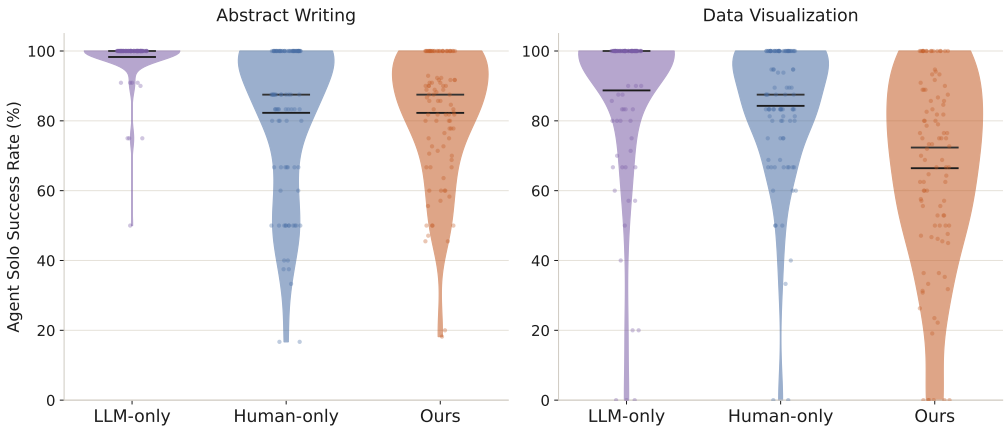


Figure 4: LLM-only rubrics fail to distinguish low-quality agent solutions (left: writing, right: data visualization), whereas our rubrics capture comparable overlooked aspects to the human-written rubrics for writing, even substantially more overlooked aspects for data visualization.

either by LLMs or humans. Yet each approach suffers from a distinct set of shortcomings. For one, LLM-created rubrics often miss aspects that human experts care about and are prone to self-bias, favoring LLM-produced solutions over ones a human evaluator would prefer (Deutsch et al., 2022; Shankar et al., 2024). However, human-only annotation is no easier, as human judgment is naturally inconsistent on open-ended tasks, and producing high-quality rubrics demands substantial effort to instruct, motivate, and supervise annotators (Xu et al., 2024a; Vidgen et al., 2026). To this end, we design our evolving rubric module to capture human expertise directly from their interaction with the agent, while relieving the manual annotation burden via automatic LLM-based rubric generation.

To show this empirically, we collect two baseline rubric sets for comparison: (i) LLM-produced rubrics given the task instruction, and (ii) human-written rubrics given the task scenario.[7] We evaluate these rubrics on capturing failures in agents' solo solutions. Since these solo solutions are later revised through numerous rounds of human iterations within the same session, we take this as evidence that they are imperfect and should score accordingly low against a truly comprehensive rubric. We evaluate agent solo solutions on each of the three rubric sets and show their scores in Figure 4.

First, scores against LLM-only rubrics saturate at an average of 98.3% and 88.7% at writing and data visualization tasks, respectively. Our evolved rubrics, by contrast, yield substantially lower average scores of 82.3% and 66.4%, suggesting they capture more failures the baseline rubrics miss. Compared to human-only rubrics, our evolved rubrics capture a comparable share of failures on writing tasks, but 17.9% more on data visualization tasks. Manually analyzing the human-only rubrics, we conjecture this gap stems mainly from a difference in annotator motivation and expertise, as we hired senior CS PhD students for the writing tasks, but general human workers for data visualization.

5 Conclusion

In this work, we introduce an adaptive agent framework that performs test-time adaptation via context and weight updates using human-agent interaction data. Throughout experiments with 20 human experts, our agents effectively adapt to individual task success within merely 20 interactions. Our evolving verifier module automatically incorporates human interaction signals and produces more comprehensive evaluation rubrics for realistic, open-ended tasks. Both adapted agents and evolved rubrics capture two layers of expertise: community expertise shared across users, and individual expertise specific to a user. Overall, our results point to a concrete design principle for future AI agents: learn continuously from test-time interaction, and adapt to the individual humans they serve. We hope this work motivates research into more advanced test-time agent adaptation techniques, as well as systematic documentation and benchmarking of community- and individual-level expertise.

[7] Given limited human resources to annotate rubrics across 100 tasks, we have humans annotate one rubric set that can be applied to all tasks in their assigned category (i.e., writing or data visualization) generally.

AI USE STATEMENT

In this work, we used generative AI tools for polishing the human writing draft and checking the validity of mathematical formulations. Additionally, we used generative AI tools for creating analysis code and performing certain scaled evaluations. We have reviewed all AI-assisted work, where the LLM-generated code was verified and tests for correctness by authors. We take responsibility for the final content of this work, including text, claims, or artifacts produced with the aid of generative AI.

REFERENCES

Tinker is a training api for researchers. https://thinkingmachines.ai/tinker/, 2025.

Marwa Abdulhai, Isadora White, Yanming Wan, Ibrahim Qureshi, Joel Leibo, Max Kleiman-Weiner, and Natasha Jaques. How llms distort our written language. *arXiv preprint arXiv:2603.18161*, 2026.

Pranjal Aggarwal, Graham Neubig, and Sean Welleck. Gym-anything: Turn any software into an agent environment. *arXiv preprint arXiv:2604.06126*, 2026.

Anthropic. Agent skills. https://agentskills.io/home, 2025.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. *arXiv preprint arXiv:2603.12273*, 2026.

Souradip Chakraborty, Jiahao Qiu, Hui Yuan, Alec Koppel, Dinesh Manocha, Furong Huang, Amrit Bedi, and Mengdi Wang. Maxmin-RLHF: Alignment with diverse human preferences. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=8tzjEMF0Vq.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Hao Zhang, Banghua Zhu, Michael Jordan, Joseph E Gonzalez, et al. Chatbot arena: An open platform for evaluating llms by human preference. *arXiv preprint arXiv:2403.04132*, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Mutlu Cukurova. What do you mean by human-ai collaboration: Prerequisite functions and the affordances needed to achieve it. *arXiv preprint arXiv:2606.15509*, 2026.

Daniel Deutsch, Rotem Dror, and Dan Roth. On the limitations of reference-free evaluations of generated text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 10960–10977, 2022.

Huanang Gao, Jiayi Geng, Wenyue Hua, Mengkang Hu, Xinzhe Juan, Hongzhang Liu, Shilong Liu, Jiahao Qiu, Xuan Qi, Qihan Ren, Yiran Wu, Hongru WANG, Han Xiao, Yuhang Zhou, Shaokun Zhang, Jiayi Zhang, Jinyu Xiang, Yixiong Fang, Qiwen Zhao, Dongrui Liu, Cheng Qian, Zhenhailong Wang, Minda Hu, Huazheng Wang, Qingyun Wu, Heng Ji, and Mengdi Wang. A survey of self-evolving agents: What, when, how, and where to evolve on the path to artificial super intelligence. *Transactions on Machine Learning Research*, 2026. ISSN 2835-8856. URL https://openreview.net/forum?id=CTr3bovS5F. Survey Certification.

Fernand R. Gobet. Expert memory: a comparison of four theories. *Cognition*, 66 2:115–52, 1998. URL https://api.semanticscholar.org/CorpusID:449552.

Robert Hawkins, Minae Kwon, Dorsa Sadigh, and Noah Goodman. Continual adaptation for efficient machine communication. In Raquel Fernández and Tal Linzen (eds.), *Proceedings of the 24th Conference on Computational Natural Language Learning*, pp. 408–419, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.conll-1.33. URL https://aclanthology.org/2020.conll-1.33/.

Yanheng He, Jiahe Jin, and Pengfei Liu. Efficient agent training for computer use. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=cDuA6ZNvCl.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *International conference on learning representations*, volume 2024, pp. 32808–32824, 2024.

Naman Jain, Jaskirat Singh, Manish Shetty, Tianjun Zhang, Liang Zheng, Koushik Sen, and Ion Stoica. R2e-gym: Procedural environment generation and hybrid verifiers for scaling open-weights SWE agents. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=7evvwwdo3z.

Liwei Jiang, Yuanjun Chai, Margaret Li, Mickel Liu, Raymond Fok, Nouha Dziri, Yulia Tsvetkov, Maarten Sap, and Yejin Choi. Artificial hivemind: The open-ended homogeneity of language models (and beyond). In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=saDOrrnNTz.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Noriyuki Kojima, Alane Suhr, and Yoav Artzi. Continual learning for grounded instruction generation by observing human following behavior. *Transactions of the Association for Computational Linguistics*, 9:1303–1319, 2021. doi: 10.1162/tacl_a_00428. URL https://aclanthology.org/2021.tacl-1.77/.

Jessy Lin, Luke Zettlemoyer, Gargi Ghosh, Wen-Tau Yih, Aram Markosyan, Vincent-Pierre Berges, and Barlas Oğuz. Continual learning via sparse memory finetuning. *arXiv preprint arXiv:2510.15103*, 2025.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

Shikhar Murty, Christopher Manning, Peter Shaw, Mandar Joshi, and Kenton Lee. Bagel: Bootstrapping agents by guiding exploration with language. *arXiv preprint arXiv:2403.08140*, 2024.

Erik Nilsson, Ian Huang, and Ryo Lu. Direct agents with visual prompts in design mode. https://cursor.com/blog/design-mode, 2026.

Tianyue Ou, Frank F. Xu, Aman Madaan, Jiarui Liu, Robert Lo, Abishek Sridhar, Sudipta Sengupta, Dan Roth, Graham Neubig, and Shuyan Zhou. Synatra: Turning indirect knowledge into direct demonstrations for digital agents at scale. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=KjNEzWRIqn.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G Patil, Ion Stoica, and Joseph E Gonzalez. Memgpt: Towards llms as operating systems. *arXiv preprint arXiv:2310.08560*, 2023.

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 *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=Cq1BNvHx74.

Chanwoo Park, Mingyang Liu, Dingwen Kong, Kaiqing Zhang, and Asuman E. Ozdaglar. RLHF from heterogeneous feedback via personalization and preference aggregation. In *ICML 2024 Workshop on Theoretical Foundations of Foundation Models*, 2024. URL https://openreview.net/forum?id=8XI7AGByAp.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simon Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, et al. Gdpval: Evaluating ai model performance on real-world economically valuable tasks. In *International Conference on Learning Representations*, volume 2026, pp. 24005–24040, 2026.

Dean A. Pomerleau. Alvinn: an autonomous land vehicle in a neural network. In *Proceedings of the 2nd International Conference on Neural Information Processing Systems*, NIPS'88, pp. 305–313, Cambridge, MA, USA, 1988. MIT Press.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Omar Shaikh, Michelle Lam, Joey Hejna, Yijia Shao, Hyundong Cho, Michael Bernstein, and Diyi Yang. Aligning language models with demonstrated feedback. In *International Conference on Learning Representations*, volume 2025, pp. 20498–20525, 2025.

Shreya Shankar, JD Zamfirescu-Pereira, Björn Hartmann, Aditya Parameswaran, and Ian Arawjo. Who validates the validators? aligning llm-assisted evaluation of llm outputs with human preferences. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pp. 1–14, 2024.

Yijia Shao, Zora Zhiruo Wang, Neel Ahuja, Yicheng Wang, Bowen Liu, and Diyi Yang. Collabskill: Evaluating human-agent collaboration on real-world tasks. *arXiv preprint arXiv:2606.09833*, 2026.

Junhong Shen, Hao Bai, Lunjun Zhang, Yifei Zhou, Amrith Setlur, Shengbang Tong, Diego Caples, Nan Jiang, Tong Zhang, Ameet Talwalkar, et al. Thinking vs. doing: Agents that reason by scaling test-time interaction. *arXiv preprint arXiv:2506.07976*, 2025.

Yuda Song, Lili Chen, Fahim Tajwar, Rémi Munos, Deepak Pathak, Drew Bagnell, Aarti Singh, and Andrea Zanette. Expanding the capabilities of reinforcement learning via text feedback. In *The 1st Workshop on Scaling Post-training for LLMs*, 2026a. URL https://openreview.net/forum?id=om12baYnKM.

Yueqi Song, Ketan Ramaneti, Zaid Sheikh, Ziru Chen, Boyu Gou, Tianbao Xie, Yiheng Xu, Danyang Zhang, Apurva Gandhi, Fan Yang, Joseph Liu, Tianyue Ou, Zhihao Yuan, Frank F. Xu, Shuyan Zhou, Xingyao Wang, Xiang Yue, Tao Yu, Huan Sun, Yu Su, and Graham Neubig. Agent data protocol: Unifying datasets for diverse, effective fine-tuning of LLM agents. In *The Fourteenth International Conference on Learning Representations*, 2026b. URL https://openreview.net/forum?id=tG6301ORHd.

Yu Sun, Xiaolong Wang, Zhuang Liu, John Miller, Alexei Efros, and Moritz Hardt. Test-time training with self-supervision for generalization under distribution shifts. In *International conference on machine learning*, pp. 9229–9248. PMLR, 2020.

Yunhao Tang, Daniel Zhaohan Guo, Zeyu Zheng, Daniele Calandriello, Yuan Cao, Eugene Tarassov, Rémi Munos, Bernardo Ávila Pires, Michal Valko, Yong Cheng, et al. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, 2024.

Rishabh Tiwari, Kusha Sareen, Lakshya A Agrawal, Joseph E Gonzalez, Matei Zaharia, Kurt Keutzer, Inderjit S Dhillon, Rishabh Agarwal, and Devvrit Khatri. Learning, fast and slow: Towards llms that adapt continually. *arXiv preprint arXiv:2605.12484*, 2026.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, et al. Apex-agents. *arXiv preprint arXiv:2601.14242*, 2026.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=ehfRiF0R3a.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Henry Wu, et al. Opencua: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025a.

Xinyuan Wang, Bowen Wang, Dunjie Lu, Junlin Yang, Tianbao Xie, Junli Wang, Jiaqi Deng, Xiaole Guo, Yiheng Xu, Chen Wu, et al. Opencua: Open foundations for computer-use agents. *Advances in Neural Information Processing Systems*, 38:139756–139806, 2026a.

Yinjie Wang, Xuyang Chen, Xiaolong Jin, Mengdi Wang, and Ling Yang. Openclaw-rl: Train any agent simply by talking. *arXiv preprint arXiv:2603.10165*, 2026b.

Zora Zhiruo Wang, Apurva Gandhi, Graham Neubig, and Daniel Fried. Inducing programmatic skills for agentic tasks. In *Second Conference on Language Modeling*, 2025b. URL https://openreview.net/forum?id=lsAY6fWsog.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Forty-second International Conference on Machine Learning*, 2025c. URL https://openreview.net/forum?id=NTAhi2JEEE.

Zora Zhiruo Wang, Yijia Shao, Omar Shaikh, Daniel Fried, Graham Neubig, and Diyi Yang. How do ai agents do human work? comparing ai and human workflows across diverse occupations. *ArXiv*, abs/2510.22780, 2025d. URL https://api.semanticscholar.org/CorpusID:282388757.

Laura Weidinger, Inioluwa Deborah Raji, Hanna Wallach, Margaret Mitchell, Angelina Wang, Olawale Salaudeen, Rishi Bommasani, Deep Ganguli, Sanmi Koyejo, and William Isaac. Toward an evaluation science for generative ai systems. *arXiv preprint arXiv:2503.05336*, 2025.

Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International conference on machine learning*, pp. 23965–23998. Pmlr, 2022.

Peng Xia, Jianwen Chen, Hanyang Wang, Jiaqi Liu, Kaide Zeng, Yu Wang, Siwei Han, Yiyang Zhou, Xujiang Zhao, Haifeng Chen, et al. Skillrl: Evolving agents via recursive skill-augmented reinforcement learning. *arXiv preprint arXiv:2602.08234*, 2026.

Frank F Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Z Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, et al. Theagentcompany: benchmarking llm agents on consequential real world tasks. *arXiv preprint arXiv:2412.14161*, 2024a.

Wenda Xu, Guanglei Zhu, Xuandong Zhao, Liangming Pan, Lei Li, and William Wang. Pride and prejudice: Llm amplifies self-bias in self-refinement. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15474–15492, 2024b.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604, 2026.

Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, et al. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

Siyan Zhao, Zhihui Xie, Mengchen Liu, Jing Huang, Guan Pang, Feiyu Chen, and Aditya Grover. Self-distilled reasoner: On-policy self-distillation for large language models. In *ICLR 2026 Workshop on Lifelong Agents: Learning, Aligning, Evolving*, 2026. URL https://openreview.net/forum?id=31wkVZXEaJ.

Boyuan Zheng, Michael Y Fatemi, Xiaolong Jin, Zora Zhiruo Wang, Apurva Gandhi, Yueqi Song, Yu Gu, Jayanth Srinivasa, Gaowen Liu, Graham Neubig, et al. Skillweaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*, 2024.

A Related Work

Test-Time Agent Adaptation Due to the persistent gap between training data and the versatility of downstream usage, adapting agents at test time has become increasingly popular for meeting ad-hoc downstream requirements (Gao et al., 2026). A large portion of this work has focused on adapting agent context. On one hand, existing studies induce procedural (Wang et al., 2025c), factual (Packer et al., 2023), and preferential knowledge into agent memory augmented at inference time. On the other hand, for further quality guarantees and efficiency, programmatic skills have been designed for robust knowledge reuse (Wang et al., 2025b; Zheng et al., 2025; Xia et al., 2026). Context-based adaptation dominates this space largely because it is easier to interpret and implement, while still yielding observable improvements from just a handful of task sessions. Partly for this reason, updating the weights of the agent backbone LM remains far less explored, constrained by limited available data as well as the difficulty of implementation and inspection (Lin et al., 2025). As a result, despite the simplicity of context adaptation, little is known about how it compares to weight-based test-time training on a per-instance basis. Recent studies introduce test-time training (TTT, Sun et al. 2020) to realize this online model weight update, leveraging weak supervision signals from experimented tasks (Yuksekgonul et al., 2026), yet still require substantial amounts of training to be effective. Our work challenges the presumed inferiority and data requirements of weight updates, demonstrating its feasibility on agentic systems beyond the backbone LMs.

Learning from Human Signals A long-standing line of work trains LMs using human binary preferences or NL feedback to improve instruction-following abilities (Christiano et al., 2017; Chiang et al., 2024; Buening et al., 2026). In agent training specifically (Pomerleau, 1988), many studies focus on training agents to imitate human demonstrations (Ross et al., 2011; Wang et al., 2025a; Shaikh et al., 2025). However, human activity typically involves UI-level interaction, which differs drastically from the programmatic approaches that LM-based agents favor (Wang et al., 2025d), making raw human demonstrations a poor fit for agent learning. To address this mismatch, some works instead have humans provide feedback on agent-driven solutions and translate that feedback into training signals (Wang et al., 2026b). While effective, these approaches typically require substantial amounts of data, which is often infeasible in real-world scenarios, overlooking many alternative action-taking signals from human activities. In contrast, our work develops an efficient adaptation strategy that meaningfully improves agent performance within just tens of task sessions.

Evaluating Agents on "Non-Verifiable" Tasks LM agents experienced a first surge of development on software engineering tasks, where execution-based, unit test-style verifiers (Chen et al., 2021) have become the standard and robust way to evaluate agent performance (Jimenez et al., 2024). However, as we move from these easily verifiable tasks to more open-ended tasks such as creative writing or visual creation, two main challenges emerge. First, we need to find alternative ways for evaluation: human evaluation is often unstable and hard-to-scale (Patwardhan et al., 2026); some studies have attempted to use LMs to directly produce a score (Liu et al., 2023), or more robustly, provide a set of binary judgments against a rubric (Aggarwal et al., 2026; Shao et al., 2026). Constructing high-quality rubrics is difficult, since for open-ended tasks, it becomes unclear what "task success" means, as there exist countless "correct" ways to solve these tasks (Jiang et al., 2026). While most works leverage LLMs or hire human annotators to construct rubrics, our experiments show that they are incomplete due to LM knowledge limitations and human lacking grounding in task execution. To address this, we propose an LLM-automated, human-in-the-loop rubric evolution strategy, to create high-quality rubrics for both model training and agent evaluation purposes.

Training Agentic Language Models Different from training a text-producing LM, agentic LMs need to consume observation-style inputs such as serialized web page content or UI-style visuals, as well as produce structured responses containing executable actions and explanatory thoughts. Beyond the proliferation of works around the low-agentic regime such as question answering or chat-style tasks (Lin et al., 2025), recent work has investigated strategies to synthesize a great volume of data (Song et al., 2026b) to train agents for common purposes such as software engineering (Pan et al., 2025; Jain et al., 2025) and web navigation tasks (Ou et al., 2024; Shen et al., 2025; Murty et al., 2024; Zhou et al., 2024). Others turn to human-sourced data such as recorded human computer-use activities (He et al., 2026; Wang et al., 2026a), yet still suggest one of the most critical recipes is constructing scalable, quality data for training. However, this is not always feasible in practice, as it

may be expensive, time- or cost-wise, to prepare numerous data for a specific downstream application. This raises an under-explored question: can we effectively update model weights by exploiting a limited number of examples? In this work, we directly tackle this problem and show that it's possible.

B Data Collection

For 20 tasks we collect human-agent interaction data on, we present their venues and topics in Table 3. In Table 4, we also list out the source of 30 held-out tasks, demonstrating their varied distributions from the 20 tasks above, and their validity in serving as our generalization test bed.

Venue	Title
NeurIPS	Artificial Hivemind: The Open-Ended Homogeneity of Language Models (and Beyond)
NeurIPS	Gated Attention for Large Language Models: Non-linearity, Sparsity, and Attention-Sink-Free
NeurIPS	1000 Layer Networks for Self-Supervised RL: Scaling Depth Can Enable New Goal-Reaching Capabilities
NeurIPS	On the Generalization Properties of Diffusion Models
ICLR	Safety Alignment Should Be Made More Than Just a Few Tokens Deep
ICLR	Learning Dynamics of LLM Finetuning
ICLR	AlphaEdit: Null-Space Constrained Knowledge Editing for Language Models
ICML	CollabLLM: From Passive Responders to Active Collaborators
ICML	Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions
ICML	Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction
ICML	Conformal Prediction as Bayesian Quadrature
ICML	Score Matching With Missing Data
ICML	The Value of Prediction in Identifying the Worst-Off
ACL	INFINI-GRAM MINI: Exact n-gram Search at the Internet Scale with FM-Index
ACL	Mind the Value-Action Gap: Do LLMs Act in Alignment with Their Values?
ACL	LINGGYM: How Far Are LLMs from Thinking Like Field Linguists?
ACL	Generative or Discriminative? Revisiting Text Classification in the Era of Transformers
ACL	Measuring Chain of Thought Faithfulness by Unlearning Reasoning Steps
CHI	Synthetic Human Memories: AI-Edited Images and Videos Can Implant False Memories and Distort Recollection
CHI	Creative Writers' Attitudes on Writing as Training Data for Large Language Models

Table 3: Titles and venues of the 20 tasks we collected human-agent interaction data on.

Venue	Title
ICLR	Transformers are Inherently Succinct
ICLR	LLMs Get Lost In Multi-Turn Conversation
CHI	Towards Fluent Interaction with Cyber-Physical Architecture
CHI	"I Don't Think RAI Applies to My Model" - Engaging Non-champions with Sticky Stories for Responsible AI Work
CHI	iTagPDF: Towards Finally Automating PDF Accessibility
NeurIPS	Visual Autoregressive Modeling: Scalable Image Generation via Next-Scale Prediction
NeurIPS	Not All Tokens Are What You Need for Pretraining
NeurIPS	The PRISM Alignment Dataset: What Participatory, Representative and Individualised Human Feedback Reveals About the Subjective and Multicultural Alignment of Large Language Models
ICLR	Generalization in Diffusion Models Arises from Geometry-Adaptive Harmonic Representations
ICLR	Never Train from Scratch: Fair Comparison of Long-Sequence Models Requires Data-Driven Priors
ICLR	Protein Discovery with Discrete Walk-Jump Sampling
ICML	Position: Considerations for Differentially Private Learning with Large-Scale Public Pretraining
ICML	Debating with More Persuasive LLMs Leads to More Truthful Answers
ICML	Genie: Generative Interactive Environments
ACL	Mission: Impossible Language Models
ACL	Why are Sensitive Functions Hard for Transformers?
ACL	Aya Model: An Instruction Finetuned Open-Access Multilingual Language Model
ACL	How Johnny Can Persuade LLMs to Jailbreak Them
EMNLP	Pretraining Data Detection for Large Language Models: A Divergence-based Calibration Method
EMNLP	An Image Speaks a Thousand Words, but Can Everyone Listen? On Image Transcreation for Cultural Relevance
EMNLP	Toward Robust Speech Representation Learning for Thousands of Languages
EMNLP	KidLM: Advancing Language Models for Children
CHI	Constrained Highlighting in a Document Reader can Improve Reading Comprehension
NeurIPS	Scaling Data-Constrained Language Models
NeurIPS	Direct Preference Optimization: Your Language Model is Secretly a Reward Model
ICLR	Emergence of Maps in the Memories of Blind Navigation Agents
ICLR	Universal Few-shot Learning of Dense Prediction Tasks with Visual Token Matching
ACL	Do Androids Laugh at Electric Sheep? Humor "Understanding" Benchmarks from The New Yorker Caption Contest
ACL	Marked Personas: Using Natural Language Prompts to Measure Stereotypes in Language Models
ACL	Weaker Than You Think: A Critical Look at Weakly Supervised Learning

Table 4: Titles and venues of the held-out tasks we tested agent generalization abilities on.

C Exploration on Weight-Based Adaptation Approaches

We conducted a broad exploration of weight adaptation approaches and selected DPO as the approach used in the main experiments (§2.3) based on its superior performance.

C.1 Alternative Algorithms

Besides DPO, we also experimented with on-policy distillation (OPD) and REINFORCE training, as formulated below.

On-Policy Distillation (OPD) applies a token-level KL divergence target from a teacher policy $\pi_{teacher}$ on the current agent policy π_θ. We adopt on-policy self-distillation (OPSD) (Zhao et al., 2026), where the teacher adopts the same parameterization θ and the teacher "privilege" comes from the additional context $\{\tau^A_{j'}, \tau^H_{j'}\}^J_{j'=j}$ it has access to, namely later iterations of agent and human activities in the session. Formally speaking, compared to the student policy $\pi_\theta(\cdot|q)$, the teacher policy is $\pi_{teacher}(\cdot|q) = \pi_\theta(\cdot|q, \{\tau^A_{j'}, \tau^H_{j'}\}^J_{j'=j})$. The learning loss can be represented as:

$$\mathcal{L}_{OPD}(\theta) = -\sum_l \sum_{t \in a_{j,l}} \pi_{teacher}(a_{j,l}|q, a_{j,<l}, s_{j,<l}) \log \pi_\theta(a_{j,l}|q, a_{j,<l}, s_{j,<l}) \tag{4}$$

$$= -\sum_l \sum_{t \in a_{j,l}} \pi_\theta(a_{j,l}|q, a_{j,<l}, s_{j,<l}; \{\tau^A_{j'}, \tau^H_{j'}\}^J_{j'=j}) \log \pi_\theta(a_{j,l}|q, a_{j,<l}, s_{j,<l}) \tag{5}$$

REINFORCE uses the scalar reward $r_j = \sum(\{r_j\})$ of the $j+1$-th agent trajectory τ^A_j, optionally with quantified human activities (e.g., number of actions $r_j = |\tau^H_j|$), as a policy gradient signal to reinforce high-reward trajectories. We represent the collective reward as $R_j = \sum(\{r_j\}) - \lambda|\tau^H_j|$, where λ is the importance factor of human activities. The overall REINFORCE loss can be formalized as:

$$\mathcal{L}_{REINFORCE}(\theta) = -(R_j - b) \sum_l \log \pi_\theta(a_{j,l}|q_j, a_{j,<l}, s_{j,<l}) \tag{6}$$

where b is the baseline reward to reduce variance, we implement b as the reward from last iteration $b = R_{j-1}$.

Performance Analysis In preliminary experiments, we find both alternatives perform worse than DPO. For REINFORCE, we find that agents rarely explore high-scoring solutions according to the evolved rubrics via human feedback, yet constantly synthesize solutions that stuck around a 0.3–0.4 success score range out of 1. We conjecture that agents have limited exposure to expert-style solutions in their training data, therefore hard to explore such solutions and impossible to hill-climb even given the rubrics without additional execution hints. In comparison, OPD has the advantage of providing privileged information to the teacher policy to hint at the student exploration. Nonetheless, at least in our data-efficient regime, we find that directly training the agent policy to prefer to generate the oracle solution (i.e., DPO) is more effective than learning the representation of it (i.e., OPD)

C.2 DPO Data Formulation

For DPO training, we compared multiple strategies of constructing pairwise data. Specifically, we experimented with (i) first-last: pairing the first and the last solution, (ii) enumerate: pairing every earlier and later solution possible, (iii) min-gap-k: pairing earlier-later solutions that are at least k iterations apart. Empirically, we find (i) performs the best, so we adopted this setup in both offline and on-policy pair construction in the main approach in §2.2.

D Human Expertise Learned During Agent Adaptation

In Table 5 and Table 6, we provide descriptions of the expertise categories shown in Figure 6, on writing and data visualization tasks, respectively.

Category	Exemplar Task
Surface form and venue conventions	Mechanical presentation rules including word count limits (150-250 words), single-paragraph layout, removal of section headings, punctuation conventions (no em-dashes), and informal register avoidance.
Active voice and direct phrasing	Preference for first-person active constructions ('we find', 'we show', 'we introduce') over passive voice, direct statements over vague constructions, and accessible terminology that prioritizes clarity for the target audience.
Terminology and naming	Handling of acronyms (expansion on first mention with parenthetical abbreviations), consistent technical naming, jargon definition or avoidance, precise use of domain terms, and elimination of ambiguous references or undefined terminology.
Problem framing	Covers how the abstract opens with the research problem or gap, establishes motivation and context.
Level of specificity	Requirements for grounded detail including specific numbers, dataset names, model scales, measurement units, sample sizes, named examples of domains or tasks, quantitative results, and concrete experimental parameters rather than abstract descriptions.
Contributions and findings presentation	What the work claims to deliver or demonstrate, including listed contributions as distinct points, headline findings with concrete support, key insights explicitly highlighted, theoretical contributions stated, and implications or takeaways for the field.
Discourse precision and compression	Combines tight, precise wording that bans hedges, vague qualifiers, and unsupported superlatives with discourse compression that enforces one idea per sentence, eliminates redundancy and repetition, and condenses filler without sacrificing clarity.
Logical flow	distinguish between different types of claims or comparisons, separate distinct ideas into their own sentences or clauses. maintains logical progression from problem to method to results with explicit transitions and narrative coherence across sentences.

Table 5: Description of observed human expertise categories in writing tasks.

Category	Exemplar Task
Delivery and HTML conventions	Mechanical packaging of the deliverable: valid self-contained HTML, required library/ CDN tags, presence of expected canvas or chart elements, and browser-renderable output without errors. Differs from Chart structure and geometry, which covers how the chart is composed once it renders.
Chart structure and geometry	Recipe of the visualization itself: chart type, orientation, number and arrangement of panels/series/groups, mark shape and sizing (bars, cells, heatmaps), grouping layout, and viewport dimensions. Differs from Delivery (file/HTML validity), Axes (numeric scales), and Data fidelity (exact encoded values).
Data fidelity	Exact match of encoded values to the specified data: cell contents, bar heights/lengths, error-bar extents, series/row/column counts, and correctly placed best-value markers. Differs from Color and visual encoding (how values are colored) and from Chart structure (layout recipe, not the numbers).
Axes, scales, and reference marks	Numeric scaffolding of the plot: axis ranges, tick intervals and order, linear vs log scaling, presence or absence of gridlines, and dashed/threshold reference lines at specified positions. Differs from Titles, legends, and labels (axis title/tick text wording) and from Data fidelity (the data values plotted on those axes).
Color and visual encoding	Mapping of values and groups onto color and related visual channels: semantic color mappings (e.g. green=desirable), colormaps and opacity, series/group color families, highlight fills, and contrast against the background. Differs from Data fidelity (whether values are correct) and from Titles, legends, and labels (legend text explaining colors).
Titles, legends, and labels	Textual and annotative content: chart/section titles, axis and series names, legend content and placement, abbreviations/naming conventions, captions, headers, plus overlay marks and affordances (significance brackets, stars/asterisks, arrows, dotted guides, hover/ tooltips). Differs from Spatial clarity (whether text/marks overlap or clip) and from Color (the color mapping itself, not the legend wording).
Spatial clarity and non-overlap	Layout hygiene so elements stay visible and distinct: label placement that avoids occlusion, no clipping of bars/error bars/tooltips, cell borders, margins/padding, and legend bounds. Differs from Titles, legends, and labels (what text/marks say or which ones exist) and from Chart structure (panel/viewport recipe rather than overlap/clipping).

Table 6: Description of observed human expertise categories in data visualization tasks.

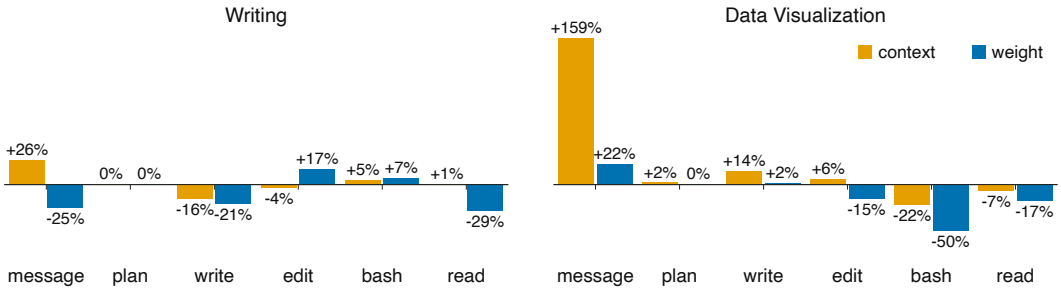


Figure 5: Relative change in mean token count by action type, for context- and weight-adaptive agents compared to the unadapted baseline agent. Positive values indicate longer action content than baseline; negative values indicate shorter action content.

D.1 Weight Adaptation Has Greater Inference-Time Efficiency than Context Adaptation

D.2 What Agents Learned, and Failed to Learn, from Humans

We analyze the passed and failed rubrics of our adapted agents against baseline and oracle, to examine what our agents learned, and failed to learn, through adaptation. Particularly, we compare test-time solutions produced by (i) the baseline agent, (ii) our adapted agents, and (iii) human-agent interaction (i.e., oracle). We measure *Agent Gains* by finding rubrics fulfilled by adapted agents (ii) yet failed by the baseline (i), and *Gaps to Oracle* with rubrics failed by adapted agents (ii) that are achieved in oracle solutions (iii). For both writing and data visualization tasks, we categorize these rubrics by themes, as where humans expressed their expertise (§D).

For writing tasks, humans expressed most of their expertise in *discourse precision and compression* (19.6%), *surface form and venue conventions* (19.1%), and *context specificity* (17.6%), reflecting practical knowledge of venue-implicit conventions that experts frequently correct agents toward, along with a persistent emphasis on precise, concise writing.

Among themes, *surface form* and *terminology and naming* are easiest to absorb, with 70.4–73.7% of expressed expertise successfully integrated, likely because they are easy to specify explicitly and straightforward for agents to adhere to. *Context specificity* and *problem framing*, in contrast, prove harder to learn, with only 41.9–43.4% of expressed expertise captured, suggesting these categories demand more implicit, situational judgment that is harder to transfer in a handful of examples. Comparing the two adaptation methods, context and weight acquire largely comparable knowledge, with similar proportions across themes, contrary to prior works that assign distinct functions to context and weights (Tiwari et al., 2026).

For data visualization, humans express expertise predominantly around textual annotations on *titles, legends, and labels* (38.5%), along with visual aspects such as *color* (15.5%) and *spatial clarity* (12.4%). Notably, humans show particular expertise in what to highlight in a figure and how to highlight it (*color and visual encoding*), where both adaptation methods exhibit a substantial gap from the human oracle, leaving 60.1% and 50.0% of expertise uncaptured. In contrast, implementation-oriented aspects are markedly easier for agents to integrate: 75.2% of *data fidelity* requirements and 94.4% of *HTML conventions* requirements are fulfilled.

E Rubrics Validation

Besides effectively capturing failures in imperfect task solutions (§4), we show our rubrics are as effective in revealing task success as other solutions, further validating the quality of our evolved rubrics.

In Figure 7, we evaluate the oracle solutions using LLM-only, human-only, and our rubrics, all of which are effective in signaling task success by producing scores around high scores. Taken together with results in Figure 4, our rubrics fail imperfect agent solutions due to issues in the solution, rather than issues in the evaluation criteria.

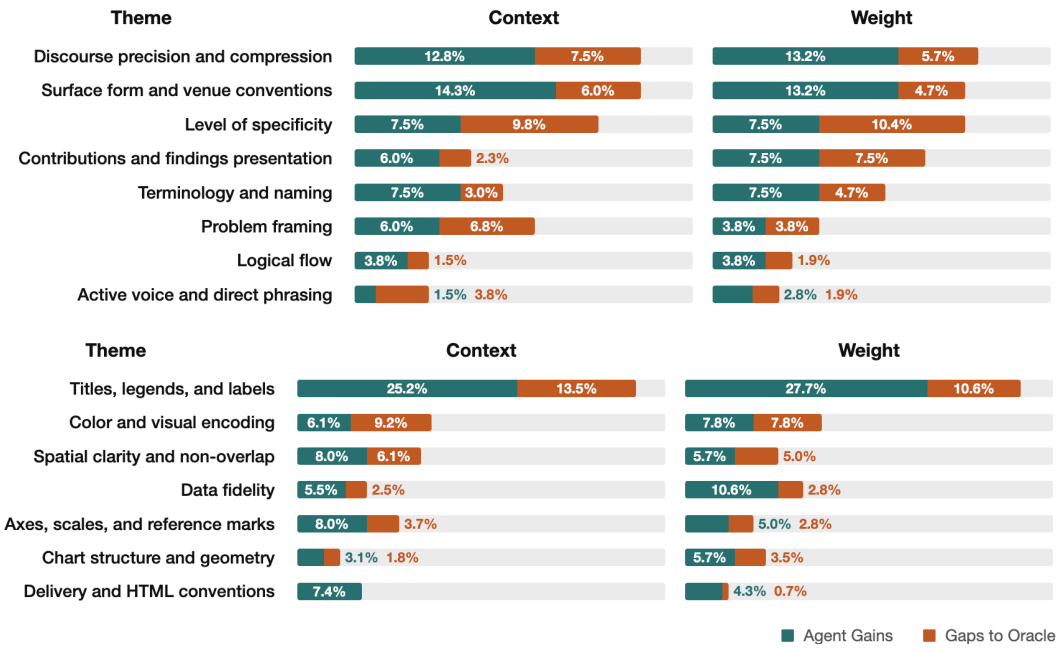


Figure 6: Expertise absorbed by agents' contexts and weights during adaptation (*Agent Gains*) and remaining from the human oracles (*Gap to Oracle*), on writing (top) and data visualization (bottom) tasks. See §D for detailed descriptions of each expertise theme.

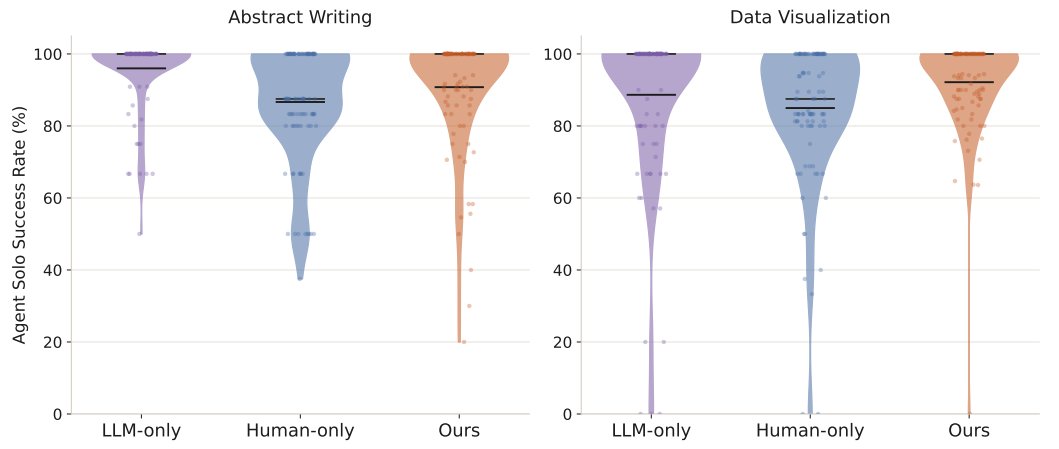


Figure 7: Our evolved rubrics are as effective in measuring task success as LLM and human-produced rubrics.

F Shared vs Personal Expertise

F.1 Rubrics Capture Shared Community Expertise and Individual Expertise

Even when interacting with the same initial agent on the same set of tasks, different humans arrive at different solutions and different rubrics. This reflects a basic property of open-ended tasks such as the writing and data visualization we investigated: they admit no single, universal notion of correctness. In fact, countless solutions can solve such a task well, that is, score highly against their corresponding evaluation criteria, depending on what the particular expert prefers. Despite this interpersonal variance, most fields still maintain community guidelines broadly agreed upon by their practitioner, forming a notion of relative correctness. We hypothesize that the evolved rubrics by

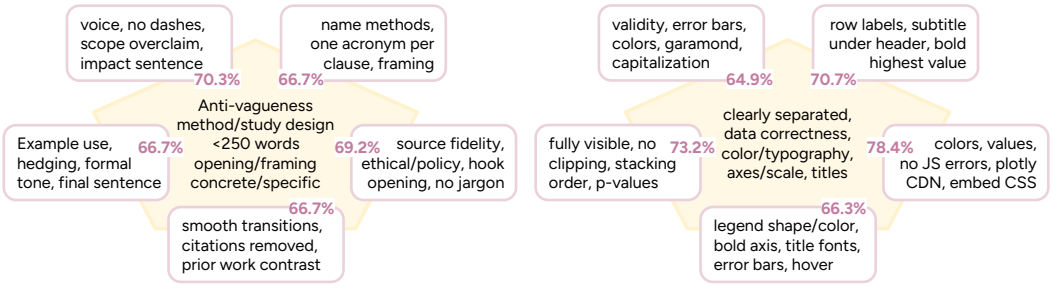


Figure 8: Shared and personalized rubrics on abstract writing (left) and data visualization (right) tasks. Each intersection value indicates the percentage of that user's rubrics classified as shared, out of all rubrics derived from their interaction.

our interface capture both components: (i) shared community guidelines, such as grounding every written claim in concrete numerical evidence; and (ii) individual strategies and preferences, such as one expert favoring enumeration while another avoids it.

To test this hypothesis, we take each user's evolved rubrics and, for every criterion, check whether it is also covered by other users' rubrics on the same task. A criterion covered by over 60% of other rubrics is categorized as a shared guideline; otherwise, we treat it as personalized. Across all writing and data visualization tasks, we find an average of 67.9% and 70.7% of user rubrics fall under shared community guidelines, with the remainder reflecting personalized expertise and preferences. Figure 8 illustrates these shared criteria alongside exemplar personalized ones. In both tasks, we observe users placing emphasis on different aspects. E.g., for visualizations, some users prioritize implementation details (e.g., Plotly CDN, CSS styling), while others emphasize statistical rigor, such as error bars and significance testing.

F.2 Agent Learns Both Shared and Personalized Expertise

A natural question follows: through this efficient adaptation, does the agent only learn individual expertise, or does it also improve on community-agreed standards more broadly? For each agent adapted to a specific expert, we separately measure its success rates on shared and personalized criteria, and compare both against the baseline agent's success rates. As shown in Table 7, beyond the significant gains on personalized criteria, agents also improve by up to 8.8% on shared criteria for the two tasks, signaling that this adaptation yields genuine gains in fundamental capability, beyond just individual fit.

Method	**Writing: ΔSR%**		**DataViz: ΔSR%**	
	Shared	Personal	Shared	Personal
Context	0.3	6.2	6.1	19.6
Weight	1.9	9.4	8.8	13.8

Table 7: Agents adapted to individual experts improve on both personalized criteria and shared community criteria, relative to the baseline agent, on writing (left) and data visualization (right) tasks.

F.3 Exploration: Deriving Community Expertise from Individual Adaptations

Although each agent is adapted towards a particular user, we've shown that an average of 69.3% of users' expertise reflects shared community guidelines. We thus ask: can individually adapted agents be consolidated to improve performance across users broadly? In this section, we explore initial strategies for merging multiple individualized agents into a single joint agent, in both context and weight channels, and demonstrate the open challenge for effective merging.

Context Merging We use an LLM (`claude-sonnet-4-6`) to merge the contexts derived from different users' interaction data into a single, shared piece of content. We then augment this shared content onto the baseline agent, and evaluate it on held-out tasks.

Weight Merging We explored two approaches to consolidate model weights derived from various users. The first, *Weight-Train*, pools all users' data to train a single model from scratch. The second, *Weight-Merge*, avoids retraining by directly merging the individually trained weights, concretely by

averaging the LoRA adapter parameters (Wortsman et al., 2022) and combine it with the base model for direct inference.

We conduct all merging with data collected from the offline scenario, since conducting a clean Weight-Train experiment is difficult with adapting agents in the online scenario. As shown in Table 8, all merging methods improve over the baseline on data visualization tasks, but yield smaller gains on writing tasks. Breaking down shared and personal rubrics, we find that on writing tasks, all merging methods satisfy personal criteria better than the baseline, yet exhibit varying performance in preserving shared criteria. Yet for data visualization tasks, variance mainly comes from meeting the personal criteria. This may be because baseline agents already perform strongly on writing, leaving less room for improvement that can generalize across users. Data visualization is comparably more open-ended, thus stressing models on divergent human expertise, whose effective integration remains an open challenge (Park et al., 2024; Chakraborty et al., 2024).

Method	**Writing: ΔSR%**			**DataViz: ΔSR%**		
	Shared	Personal	Overall	Shared	Personal	Overall
Context	-4.4	1.3	-2.7	2.9	-0.5	+1.1
Weight-T	-4.7	3.5	-2.3	3.6	2.7	+3.3
Weight-M	0.0	3.2	+1.0	3.6	1.8	+3.1

Table 8: Exploring different approaches to consolidate individualized agents into one.

Another potential factor is the offline nature of our adaptation: compared with online adaptation, where the agent receives human feedback targeted at its current state, offline adaptation uses potentially outdated human feedback thus leads to smaller gains (Tang et al., 2024). Together, these findings highlight the challenge of effectively bringing versatile individual expertise into a single agent and motivate future work for expertise consolidation.

G Prompts for LM-Supported Modules

Prompt for Memory Induction for Context-Adaptive Agents

Merge new induction entries into the existing memory file and produce a refined, cross-session version.
You receive (i) the original memory file and (ii) new entries derived from a new session. Merge (ii) into (i): keep durable prior preferences, add genuinely new ones, and lightly deduplicate. Output line count should stay about the same as the original memory (i). Do not add excessive new entries.
Keep: - Specific user preferences that apply to certain contexts - Recurring styling or workflow habits (e.g. larger fonts, no gridlines, compact layout) across tasks - General facts about how the user works
Remove or merge only when necessary: - Nonsensical or contradictory entries - Highly task-specific details unlikely to help elsewhere (e.g. a color for one named column/bar) - Duplicate lines that say the same thing in different words. - Make each line concise but do not over-compress away useful detail.
Output rules: - One entry per line. Plain text only (NO markdown headers like # or ##). - Prefix each line with "Fact:" or "Preference:". - Do not include reasoning or a thinking process.
Reply with: Title: <short topic name> - Fact: <item> - Preference: <item>

Prompt for Skill Induction for Context-Adaptive Agents

From the task and numbered log, describe the workflow the agent used: ordered steps, generalized (no long paths).
Your primary job is to capture what THIS session did—especially techniques, fixes, and steps visible in the Log. The existing skill file—if any—is background only.
When an existing skill file is provided: - FIRST update the workflow using concrete steps from this session's Log (mandatory when the log is non-empty). - Add or revise steps for anything new in this session (e.g. chart tweaks, file edits, verification, user-requested changes). - Adopt the useful parts from original steps only when still accurate; merge duplicates. - Do not return the existing skill unchanged if the log shows new agent work. Output the full updated skill (not a diff). Generalize: no long paths, file paths, or raw code.
Reply with: Title: <short task name> 1. <step> 2. <step> ... If nothing fits: NONE

Prompt for Grading Solutions Against Rubrics

You are an automated checker for completed task output files. Given verifier criteria and current output files, decide whether each criterion is satisfied. Reply with ONLY a JSON object of this exact shape: "results":["pass":true,"pass":false,...] Task Instruction
Verifier criteria (in order): Criteria 1 Criteria 2 ...
Output files and contents: File Content

Prompt for Summarizing Test-Time Task Rubrics

You distill grading rubrics for task category tasks.
You are given: (A) High-quality, task-specific verifiers written/edited by a user across many past tasks. (B) A fixed list of human-written general guidelines that every visualization should satisfy.
Your job: produce a JOINT GENERAL verifier list for this expertise overall.
Rules: - Capture recurring themes from the user's history (layout, labels, legends, annotations, spacing/overlap, axes/scales, colors, HTML validity, browser renderability, chart-type-specific checks when they recur). - Phrase items as general, reusable criteria (not tied to one past task's numbers/labels). - Do NOT include criteria whose only job is checking that plotted numeric values match an instruction (those are task-specific). Prefer structural/visual quality checks. - MUST cover every human guideline in (B). Either keep them (possibly tightened) or rewrite them so the same requirement is clearly present. Do not drop a human guideline. - Prefer the user's concrete, falsifiable style from (A) over vague wording. - Deduplicate near-duplicates; keep one clear criterion per idea. - Typically 10–20 items. Prefer fewer clear lines over a long union.
Reply with JSON only (no markdown fences, no commentary): "general_verifiers": ["...", "..."], "human_coverage": ["human": "<exact human guideline>", "covered_by": "<matching general_verifiers item>"] Every human guideline must appear exactly once in human_coverage.

Prompt for Determining Shared Rubrics

You compare grading rubrics for the same abstract-writing task.
You are given: (A) Source rubrics — criteria from one user's verifier list for this task. (B) Target rubrics — criteria from another user's verifier list for the SAME task.
For EACH source rubric, decide whether it is COVERED by the target list: - covered=1 if at least one target rubric checks the same requirement (exact match, paraphrase, or a target rubric that subsumes the source check). - covered=0 if no target rubric addresses that requirement.
Rules: - Match on meaning, not wording. "under 250 words" covers "Word count < 250". - A single target rubric may cover multiple source rubrics, and vice versa. - Partial overlap that still enforces the core constraint counts as covered. - Do not invent coverage from vague thematic similarity alone. - Return one judgment per source rubric, in the same order as (A).
Reply with JSON only (no markdown fences, no commentary): "coverage": ["source_index": 0, "source": "<exact source rubric text>", "covered": 1, "covered_by": "<exact target rubric text, or empty string if covered=0>", "note": "<brief reason>"] Every source rubric must appear exactly once, in order, with covered as 0 or 1.