Automating SKILL.md Generation for Computer-Using Agents via Interaction Trajectory Mining

Yuexing Hao [1] **Xiaomin Li** [2]

Abstract

Explicit skill libraries make computer-using agents easier to inspect, but hand-written `SKILL.md` files are costly to maintain. We study whether they can be mined from interaction trajectories and used as training signal. Our pipeline segments GUI trajectories, clusters segments into candidate skills, and trains a skill-aware policy. The mined clusters are readable on the source benchmark: five of eight clusters have at least 0.95 purity against InteraSkill Workflows labels. However, readability does not imply transfer. GRPO improves IW skill-step accuracy only from 18.5% to 20.5%, leaves BrowseComp+ essentially unchanged, and underperforms simple frequency priors. Trace diagnostics show three failure triggers: non-portable action-jump thresholds, order-less segment representations, and an offline IW reward model that does not yield cross-domain skill composition. Trajectory mining exposes inspectable structure, but this pipeline is not yet a reliable policy intervention.

1. Introduction

Computer-using agents (CUAs) act on graphical user interfaces (GUIs) by clicking, typing, scrolling, copying, and pasting (Yao et al., 2022; Deng et al., 2023; Zhou et al., 2024). As these agents move from single-step web tasks to longer workflows, repeated patterns become important. A user may need to search for a page, copy a value, switch applications, fill a form, or send a message. Agent systems often package these repeated routines as skills: named procedures that sit above primitive user-interface (UI) actions and below full task plans. We use `SKILL.md` to refer to this kind of explicit skill file. These files make behavior easier to inspect and debug, but they are usually written by hand.

Hand-written skills create a practical bottleneck: they must be named, scoped, documented, and updated as interfaces change. Trajectory data offers another source of structure, since repeated action subsequences may reveal natural skill units. This paper asks whether such structure can become explicit skill libraries that help a policy on new tasks. We therefore ask: *what transfers from mined skills?*

Our pipeline has three steps. First, we cut trajectories at large action changes. Second, we cluster the resulting segments and refine the cluster embedding with pseudo-label contrastive learning. Third, we train Qwen3-8B with GRPO from the base model. The reward model scores full skill-aware responses, but the main reported Phase 3 evaluation is skill-sequence composition: predicting the next mined skill labels and comparing the resulting skill sequence with the reference sequence. We compare this policy to zero-shot baselines with the same skill-oriented format on IW, WebArena, and BrowseComp+. We additionally report WorkArena-NLP as a text-only diagnostic and Mind2Web task-completion checks as context, but neither is used to claim current GRPO transfer.

The main results identify a reproducible failure pattern. The mined IW library is readable, but weak as downstream training signal: GRPO raises IW skill-step accuracy only from 18.5% to 20.5%, while BrowseComp+ changes from 43.5% to 43.3%. A Frequency baseline is stronger than the proposed MLP and GRPO policies on IW, and has lower edit distance than Auto-`SKILL.md` at every data size. Trace diagnostics show why: boundary triggers fire inside skills, clusters lose action order, and the policy learns a reward proxy that does not improve held-out composition.

We make three contributions. First, we define stage-specific failure boundaries for segmentation, taxonomy transfer, composition, and intervention against simple priors. Second, we provide trace diagnostics: threshold sweeps, action-type profiles, per-skill accuracy, exact-match checks, and data-efficiency curves. Third, we report a negative result with repair evidence: oracle retuning can repair WebArena segmentation, supervised sequence models outperform GRPO

[1]Massachusetts Institute of Technology, Cambridge, MA, USA
[2]Harvard University, Cambridge, MA, USA. Correspondence to: Yuexing Hao <yuexing@mit.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

under another modality, and generated `SKILL.md` files beat a manual table but not the Frequency baseline. Our reproducible codes are available in the anonymous Github repository [1].

2. Related Work

Modern CUAs use structured UI observations and fixed action spaces. WebShop (Yao et al., 2022) and Mind2Web (Deng et al., 2023) define common primitives, while WebArena (Zhou et al., 2024), VisualWebArena (Koh et al., 2024), WorkArena (Drouin et al., 2024), and OS-World (Xie et al., 2024) test realistic web and operating-system tasks. Foundation CUA systems such as OpAgent (Guo et al., 2026), OpenCUA (Wang et al., 2025), and UltraCUA (Yang et al., 2025) push this direction with larger models and trajectory corpora. Our emphasis on inspectable skill files also connects to human-centered and rule-grounded AI work outside GUI-agent benchmarks. Hao et al. study AI systems for shared decision-making with older adult cancer patients (Hao et al., 2024), EHR-integrated LLM agents for prostate-cancer patient education (Hao et al., 2025b), and physician–AI relevance alignment in medical question answering (Hao et al., 2025a); Li et al. study guideline-following medical decisions, rule-based data selection, and adaptive safety rules for reward modeling (Li et al., 2025a; 2024; 2025b). These systems motivate the same design principle we adopt here: automated agents should expose intermediate structure that humans can inspect, question, and correct.

Prior work already mines or synthesizes reusable workflow artifacts. Agent Workflow Memory (AWM) (Wang et al., 2024) induces routines from trajectories, SkillWeaver (Zheng et al., 2025) distills website practice into reusable API-style skills, AutoManual (Chen et al., 2024) constructs environmental manuals, ICAL (Sarch et al., 2024) distills demonstrations into cognitive abstractions, LearnAct (Liu et al., 2025) studies demonstration-based mobile GUI agents, and Open-World Skill Discovery uses action-prediction error for boundary detection (Deng et al., 2025). The broader reinforcement-learning literature gives the formal background for temporal abstraction: options and option-critic methods (Sutton et al., 1999; Bacon et al., 2017), deep HRL systems (Kulkarni et al., 2016; Vezhnevets et al., 2017; Nachum et al., 2018; Riemer et al., 2018; Li et al., 2019), and meta-learned or offline primitives (Frans et al., 2018; Ajay et al., 2021). Unsupervised skill-discovery methods such as VIC, DIAYN, DADS, CIC, and actionable representations learn reusable behaviors through mutual-information, contrastive, or goal-conditioned objectives (Gregor et al., 2017; Eysenbach et al., 2019; Sharma et al., 2020; Laskin et al., 2022; Ghosh et al., 2019); recent analysis cautions that mutual-information skills are not universally optimal for every downstream reward (Eysenbach et al., 2022). Our negative results are consistent with this caution: a coherent skill space is not automatically a useful cross-domain policy.

Recent GUI-agent training work informs our GRPO setup. DigiRL (Bai et al., 2024) and WebRL (Qi et al., 2025) optimize agents with online RL curricula; AgentTrek (Xu et al., 2025) and OS-Genesis (Sun et al., 2025) generate agent trajectories; Proposer-Agent-Evaluator (PAE) (Zhou et al., 2025) uses evaluator feedback for autonomous skill discovery; and Skills-Coach (Tian et al., 2026) applies a GRPO-style loop to generated task suites. Our experiment is narrower: it uses an offline IW-derived reward over text skill plans, does not interact with a live GUI during RL, and does not train the reward on target-domain task success. We therefore interpret weak GRPO transfer as a pipeline-level result, not as evidence against GUI-agent RL broadly.

3. Problem Setup

We treat a trajectory as a sequence of UI observations and primitive actions. Primitive actions include click, type, scroll, copy, and paste. A skill label summarizes a contiguous action segment. Formally, the input dataset is

$$\mathcal{D} = \{\tau^{(n)}\}_{n=1}^{N}, \qquad \tau^{(n)} = ((o_1, a_1), \ldots, (o_T, a_T)), \tag{1}$$

where o_t is a GUI observation and $a_t \in \mathcal{A}_{\text{low}}$ is a primitive UI action. The goal is to induce a skill vocabulary $\mathcal{Z}$ and a segmentation of each trajectory into contiguous intervals, each assigned one skill $z \in \mathcal{Z}$. In hand-authored systems, $\mathcal{Z}$ is provided by designers. In our setting, $\mathcal{Z}$ is induced from trajectories.

The question is whether the induced vocabulary $\mathcal{Z}$ helps the policy. We therefore focus the main evaluation on skill composition: whether a model can choose the right mined skill sequence on held-out tasks and transfer settings. Primitive UI-action accuracy is reported only for the Mind2Web diagnostic, where benchmark annotations directly support that metric; it is not the primary Phase 3 claim.

Failure boundaries. We use "failure" operationally: false internal cuts or poor threshold transfer; readable IW clusters that do not form a portable vocabulary; no improvement over zero-shot or simple non-LLM baselines; or a repair that helps one comparison but not the strongest trivial control.

4. Method: Automated `SKILL.md` Generation

The pipeline has three stages. It segments trajectories, clusters the segments into skills, and trains a CUA policy with

[1] Anonymous project repository: https://anonymous.4open.science/r/CUA-1680.

the resulting annotations. The first two stages build the skill library. The third stage tests whether the library helps. Figure 1 summarizes the study design. The equations below are operational definitions rather than standalone theoretical claims. Equation 2 decides where candidate skills begin and end; Equation 3 turns each variable-length segment into a fixed-length vector; Equation 4 turns those vectors into a distance matrix for clustering; and Equation 5 refines the resulting pseudo-labels into embeddings used by the sequence models. Methodologically, the pipeline combines simple offline change-point detection ideas (Truong et al., 2020), Gaussian optimal-transport geometry (Dowson & Landau, 1982; Peyré & Cuturi, 2019; Cuturi, 2013), and supervised contrastive representation learning (Khosla et al., 2020). We treat each stage as both a proposed intervention and a possible failure trigger: the boundary rule can over-split, the representation can erase order, and the reward model can optimize an IW proxy that does not transfer.

4.1. Phase 1: Trajectory Segmentation (Skill Boundary Detection)

Given an action trajectory, we use adjacent action distance as a cheap change-point signal. For a trajectory $(a_0, a_1, \ldots, a_T)$, we compute:

$$\Delta a_t = \|a_t - a_{t-1}\|_2, \quad t \in \mathcal{B} \text{ if } \Delta a_t > \theta, \tag{2}$$

where θ is selected on held-out IW data by sweeping empirical percentiles of Δa_t and maximizing boundary F1. The boundary set $\mathcal{B}$ splits a trajectory into candidate skill segments. Each action vector has 15 normalized features: a 10-way primitive-action one-hot vector, screen coordinates $(x, y) \in [0, 1]^2$, normalized timestamp, clipped text length, and clipped scroll amount. The Euclidean score is unweighted over these features and does not use DOM, screenshot, accessibility-tree, or language state. For transfer, we apply the IW-derived threshold directly; target-domain threshold sweeps are reported only as oracle diagnostics. On IW, F1 is stable over the 40th–60th percentile range (Appendix Table A6).

The rule is intentionally simple, but it can over-split skills: click-to-type transitions, typing-length changes, and large coordinate jumps can exceed θ even when the user remains inside one skill. A natural replacement would learn boundaries from action-prediction error, as in skill-boundary detection for unsegmented demonstrations (Deng et al., 2025); we leave that comparison for future work.

4.2. Phase 2: Skill Embedding (Skill Library Construction)

Step 1: Segment Representation. Let segment τ_i contain T_i action vectors $a_{i,1}, \ldots, a_{i,T_i} \in \mathbb{R}^d$. We summarize the segment by the mean and diagonal variance of its action vectors:

$$\begin{aligned} \mu_i &= \frac{1}{T_i} \sum_{t=1}^{T_i} a_{i,t}, \\ \Sigma_i &= \operatorname{diag}\left(\frac{1}{T_i} \sum_{t=1}^{T_i} (a_{i,t} - \mu_i) \odot (a_{i,t} - \mu_i) + \epsilon \mathbf{1} \right), \end{aligned} \tag{3}$$

where $\odot$ denotes element-wise multiplication and $\epsilon = 10^{-4}$ prevents degenerate variances. This is a length-invariant bag-of-actions summary. It is not a Gaussian-mixture model and it does not preserve within-segment order. This choice is useful for cheap clustering because segments with similar action inventories are close even when they have different lengths, but it discards the sequential structure that makes many GUI skills executable: selecting before copying, copying before pasting, opening a menu before choosing an item, or navigating before filling a form. Thus a cluster can be readable while still missing the order information needed for reliable downstream composition. Despite this simplification, the representation produces clusters that align with IW ground-truth skills (Section 6.2); the downstream sequence results test whether that bag-of-actions abstraction is sufficient for policy learning.

Step 2: Wasserstein Clustering. Let $v_i = \operatorname{diag}(\Sigma_i)$ denote the vector of diagonal variances. We group similar segments using the squared Bures distance between the diagonal Gaussians $\mathcal{N}(\mu_i, \Sigma_i)$:

$$\begin{aligned} D(\tau_i, \tau_j) &= \|\mu_i - \mu_j\|_2^2 + \left\|\sqrt{v_i} - \sqrt{v_j}\right\|_2^2 \\ &= \|\mu_i - \mu_j\|_2^2 + \sum_{k=1}^{d} \left(\sqrt{v_{i,k}} - \sqrt{v_{j,k}}\right)^2, \end{aligned} \tag{4}$$

This is the closed-form squared 2-Wasserstein distance for diagonal Gaussian summaries, a special case of the Fréchet/Bures distance between Gaussian measures (Dowson & Landau, 1982; Peyré & Cuturi, 2019). Because v_i stores variances, the covariance term compares standard deviations, not raw variance vectors; using $\|v_i - v_j\|_2^2$ would be a different heuristic and is not the intended metric. The implementation follows Equation 4 by taking square roots of the diagonal variances before computing the covariance distance. We run average-linkage agglomerative clustering on this distance matrix and sweep k from 8 to 16.

Step 3: Supervised-Contrastive Refinement. Let $f_\theta : \mathbb{R}^{2d} \to \mathbb{R}^{d_{\text{skill}}}$ be an MLP encoder mapping a segment summary $x_i = [\mu_i; \operatorname{diag}(\Sigma_i)]$ to an ℓ_2-normalized embedding $z_i = f_\theta(x_i)$. Using the Wasserstein cluster assignments c_i as *pseudo-labels*, we train f_θ with the supervised-contrastive

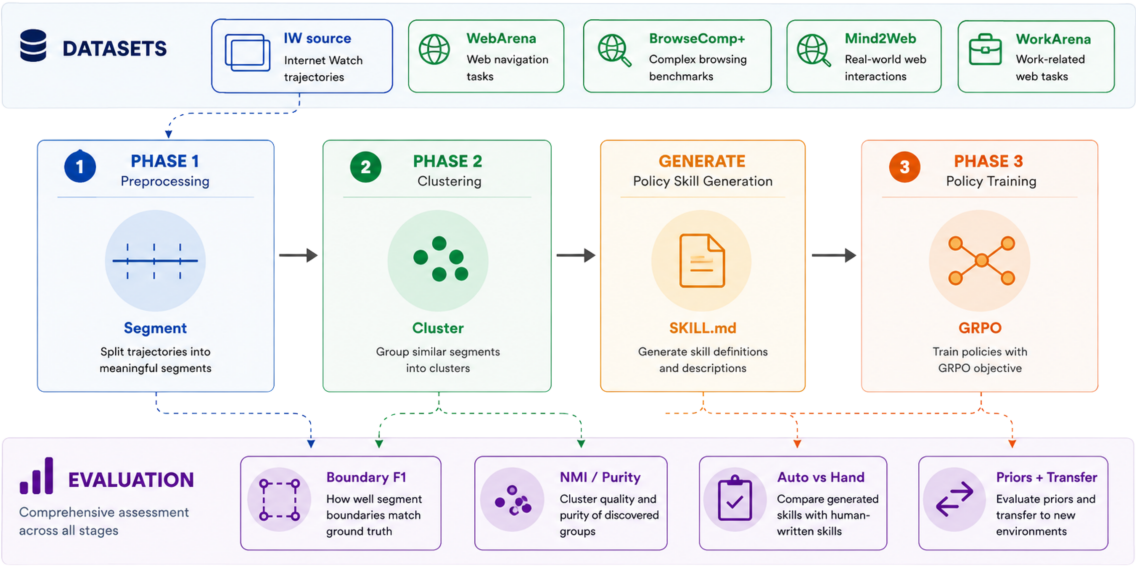

Figure 1. Study design for automated `SKILL.md` generation. IW is the source dataset for trajectory segmentation, skill-library construction, and Phase 3 GRPO policy training; WebArena and BrowseComp+ are the completed held-out transfer checks. Mind2Web and WorkArena-NLP are reported only as diagnostics, not as current GRPO transfer evidence. The paper evaluates boundary quality, cluster quality, auto-generated versus hand-crafted `SKILL.md` files, simple priors, and completed transfer checks.

loss (Khosla et al., 2020):

$$\mathcal{L}_{\text{sup-con}} = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \frac{-1}{|P(i)|} \sum_{p \in P(i)} \log \frac{\exp(z_i^\top z_p / T)}{Z_i}, \qquad Z_i = \sum_{a \in \mathcal{B} \setminus \{i\}} \exp(z_i^\top z_a / T), \tag{5}$$

where $\mathcal{B}$ is a mini-batch, $P(i) = \{p \in \mathcal{B} \setminus \{i\} : c_p = c_i\}$ is the set of same-pseudo-label positives for anchor i, $T = 0.07$ is a temperature, and the denominator sums over all other batch elements. We use a class-balanced sampler so each batch contains roughly equal members per pseudo-label.

The encoder is an MLP over $D_a = 15$ action-feature dimensions, so its input is the 30-dimensional vector $[\mu; \text{diag}(\Sigma)]$. It maps $30 \to 64 \to 32 \to d_{\text{skill}} = 16$ with ReLU activations and an ℓ_2-normalized output. We train for 200 epochs with AdamW (lr=10^{-3}, weight decay 10^{-4}), batch size 256, temperature $T = 0.07$, and an 80/20 stratified train/validation split by pseudo-label. The IW run takes less than five minutes on CPU.

Pseudo-supervision, not self-supervision. Step 2 assigns labels by clustering. Step 3 trains an encoder to predict those labels through a supervised-contrastive objective. No human skill labels enter the clustering, but the encoder is still trained with pseudo-labels. This could simply overfit to the initial clusters. We test this with random labels and a k-means baseline in Section 6.2. Wasserstein pseudo-labels perform better than both.

4.3. Phase 3: Skill-Aware GRPO Training

After mining the skill library, we train models for skill-sequence composition. The lightweight baselines test whether the mined representation is useful without an LLM: an MLP receives the current skill embedding plus sequence position and predicts the next skill, while a 2-layer causal Transformer ($d_{\text{model}} = 64$, 4 heads) attends over the skill sequence to test whether sequence context helps. The main policy is Qwen3-8B trained with GRPO from the base model on 1,275 prompts containing task context and mined skill names; its response format includes a skill label and a next-action field, but the reported IW/WebArena/BrowseComp+/WorkArena-NLP metrics evaluate the predicted skill sequence. A learned trajectory reward model scores the full prompt-response pair, and GRPO updates the base policy directly with no supervised warm start. We use 8 candidate responses per prompt at temperature 0.7, maximum completion length 192, learning rate 5×10^{-6}, gradient accumulation 8, reward clipping at 5.0, and one training epoch. The completed run took 6,072 seconds on 4 NVIDIA H200 NVL GPUs with 143,771 MiB memory per GPU.

4.4. Transfer Evaluation

We evaluate transfer with held-out benchmarks primarily through the IW-derived skill vocabulary. The claimed Phase 3 comparison covers IW as the source benchmark and WebArena/BrowseComp+ as held-out transfer checks. For Mind2Web, we separately report task-completion, action-accuracy, and element-accuracy diagnostics, but we do not use those numbers as evidence that the current GRPO policy improves primitive UI-action prediction. WorkArena-NLP is reported only as an auxiliary text-only planning diagnostic.

5. Experimental Setup

We evaluate the claimed pipeline on one source benchmark and two held-out transfer checks. IW is the source distribution: 2,000 synthetic enterprise-style trajectories with explicit skill boundaries and labels, used for segmentation, clustering, and policy training. WebArena and BrowseComp+ test whether the learned skill vocabulary transfers to external browsing traces. We also include two non-claim diagnostics: Mind2Web for web-action context, and WorkArena-NLP, a text-only conversion of WorkArena task schemas into natural-language goals and structured JSON targets. Live WorkArena results are not part of the claimed evaluation in this submission. Appendix B gives dataset sizes, benchmark status, and diagnostic details.

For Phase 3 skill composition, the main comparison is Qwen3-8B zero-shot versus Qwen3-8B trained with GRPO from the base model. We also include zero-shot Llama-3.1-70B, DeepSeek-R1-Distill-Qwen-32B, Gemma-4-31B, OLMo-3-7B, GPT-5, Claude Sonnet 4.5, and Claude Haiku 4.5 where available, plus simple non-LLM baselines. Segmentation is scored by boundary precision, recall, and F1; clustering by NMI, silhouette, and purity; skill composition by per-step accuracy, exact sequence match, and normalized edit distance; and Mind2Web by task completion, partial task success, skill accuracy, and action accuracy. Additional model, metric, and hyperparameter details are in Appendix B and Appendix A.

6. Failure Analysis Results

6.1. Failure Mode 1: Action Jumps Trigger Boundary Over-Splitting

The first failure trigger is minimal: apply an adjacent-action ℓ_2 threshold to a GUI trace. Equation 2 asks whether large adjacent action changes mark true skill switches. On IW, the best threshold is $\theta = 1.545$ (50th percentile), giving precision 0.419, recall 0.803, and F1 0.538 (Appendix Figure A2). High recall means most true skill boundaries do create visible action jumps. Low precision exposes the failure boundary: the same jumps also happen inside a skill when users click before typing, move between distant UI elements, scroll during review, or paste text as one step of a longer data-transfer routine. The detector therefore discovers many real transitions, but its trace-level failure is over-splitting ordinary within-skill behavior.

The attempted repair clarifies the mechanism. On WebArena, applying the IW-derived threshold $\theta = 1.545$ gives precision 1.000, recall 0.100, and F1 0.119. A target-domain oracle sweep can choose $\theta = 0.603$ and reach F1 0.851, but that uses WebArena boundary labels and is not a valid zero-shot transfer result. The oracle result shows that WebArena map-navigation trajectories contain clear discontinuities under their own action scale. It also shows what does *not* fix the failure: an IW-calibrated boundary rule is not a portable trigger, and target-domain retuning only works when the evaluator already has labels.

6.2. Failure Mode 2: Readable Skills Remain Source-Bound

The main positive result is narrower than expected: trajectory mining produces readable source-domain skills, but the structure does not automatically become a portable vocabulary. Equation 3 compresses each discovered segment into a mean/variance action profile, and Equation 4 clusters those profiles. With the Bures metric, agglomerative clustering reaches NMI = 0.650 at $k = 8$ on IW (Appendix Table A7). NMI drops for larger k, while purity stays near 0.63, so we use $k = 8$ in the main analysis.

The pseudo-label contrastive encoder strengthens this source-domain structure. After 200 epochs, KMeans in the 16-dimensional latent space reaches NMI = 0.862, silhouette = 0.554, and purity = 0.837, a 33% relative NMI gain over the Wasserstein baseline. Appendix Figure A3 shows the t-SNE projection, and Appendix C.5 reports the training curve. This result shows that the cluster-derived pseudo-labels are internally consistent enough to train an embedding model.

The failure boundary is transfer. WebArena looks very different: with the IW threshold, segmentation F1 falls to 0.119, and the learned embedding is weak (NMI = 0.049, silhouette = -0.255). The target-tuned WebArena threshold can give F1 = 0.851, but that is an oracle diagnostic, not a transferable setting. The map-only trajectories do not contain enough skill diversity to form a rich library, and their discontinuity scale differs from IW. The mined taxonomy is therefore best read as a source-domain discovery result, not as evidence of a general GUI skill vocabulary.

Table 1 explains why the result is tempting but insufficient. Five of eight clusters have purity at least 0.95 against one IW skill. Their action profiles are also inter-

pretable: `document_edit` contains click/format/save, `data_transfer` contains click/copy/paste, and `organize_files` contains click/right-click patterns. The weaker clusters, such as `send_message` and `review_content`, absorb broad click/type/scroll routines shared by several IW skills. The source-domain taxonomy is readable, but readability does not imply downstream usefulness. Appendix Figure A4 provides the full action-distribution heatmap.

6.3. Failure Mode 3: Learned Composition Does Not Beat Simple Priors

The strongest negative result appears after the readable clusters are used for prediction. The MLP and Transformer consume the learned segment embeddings z_i, while the GRPO policy consumes text prompts built from the same mined skill labels. Appendix Table A3 compares these pipeline variants on IW, but it is not a controlled architecture-only or reward-model ablation: input modality, model class, supervision signal, and optimizer all change at once. The comparison is still useful as a sanity check, because every variant is supposed to benefit from the mined skill structure.

The sanity check fails for the main learned policy. The GRPO-trained Qwen3-8B policy reaches 20.5% skill-step accuracy, below the MLP baseline (23.3%), the Transformer baseline (34.6%), and the trivial Frequency baseline in Appendix Table A2 (34.9%). Exact sequence match is 0% for the GRPO policy. This gives a falsifiable failure boundary: a mined-skill intervention has not repaired composition unless it beats simple class-imbalance and transition priors. The result should be interpreted as a failure of the current text-prompted GRPO setup as a whole, not as proof that the reward model alone is responsible. A reward-specific conclusion would require a modality-matched control, such as Qwen3-8B trained on the same text prompts with supervised next-skill loss, or an embedding-based policy trained with and without the same reward model.

One likely reason is that the useful part of a GUI skill is often ordered, while the segment embedding is not. The Phase 2 representation summarizes a segment by mean and variance, so it does not record whether a click happened before or after a type, copy, paste, or menu action. Per-skill accuracy follows this pattern: it ranges from 94.1% for `search_navigate` to 55.6% for `presentation_edit` (Table A8). Skills with distinct action signatures are easier to predict; skills that share click/type/scroll routines are harder. Appendix Figure A1 shows the per-position MLP and Transformer comparison.

6.4. Failure Mode 4: GRPO Does Not Repair Transfer

Table 2 gives the completed benchmark-level view. IW is the source benchmark; WebArena and BrowseComp+ are the held-out transfer checks used for the paper's policy-transfer claim. WorkArena-NLP and Mind2Web are included in the same table only as auxiliary diagnostics: WorkArena-NLP is text-only planning, while Mind2Web reports task completion on 200 `test_task` web tasks. The API models are zero-shot format-constrained baselines. The current GRPO run improves IW skill-step accuracy only slightly over zero-shot Qwen3-8B (18.5% to 20.5%), decreases on WebArena (55.8% to 44.2%), is unchanged on BrowseComp+ (43.5% to 43.3%), matches zero-shot Qwen3-8B on WorkArena-NLP field accuracy (37.0% for both, with 0% exact match), and slightly decreases Mind2Web task completion (9.5% to 9.0%). The repair therefore fails under the completed transfer checks: optimizing the IW-derived reward model does not produce a robust cross-domain skill-composition improvement.

This is surprising because the training signal is built around the mined skills, but the verified transfer gains are absent or negative. Closed-source models improve the prompting baseline but do not dominate every benchmark: GPT-5 and Claude Sonnet 4.5 are strongest on WebArena, OLMo-3-7B remains strongest on BrowseComp+, and DeepSeek-R1-Distill-32B has the highest completed Mind2Web TCR. WorkArena-NLP completed runs range from OLMo-3-7B at 12.8% to GPT-5 at 40.6%, with DeepSeek-R1-Distill-32B at 31.2% and both Qwen3-8B variants at 37.0%; these numbers diagnose text-only schema recovery, not live enterprise automation. Appendix Table A2 reinforces the negative result: a trivial Frequency baseline is strong on IW, strictly outperforming both the MLP and GRPO policies on skill-step accuracy. The current learned skill-composition methods have therefore not shown value beyond dataset class imbalance.

The Mind2Web diagnostic is not used as a policy-transfer claim because it mixes skill prediction and primitive action grounding. It is still useful as a sanity check: the current GRPO run does not beat zero-shot Qwen3-8B on end-to-end task completion, and the newly completed API runs range from GPT-5 at 0.0% to Claude Haiku 4.5 at 20.0%.

6.5. Intervention Check: Auto-`SKILL.md` Beats a Manual Table but Not Frequency

The final result concerns the artifact that motivated the pipeline: generated `SKILL.md` files. We run the system on IW subsets and generate skill descriptions, transition probabilities, workflows, and error-handling patterns, then compare these files with hand-written baselines. Appendix D.2 gives three same-task qualitative comparisons.

Figure 2 and Appendix Table A9 compare generated and hand-written `SKILL.md` files as the number of IW training trajectories changes. The important question is not whether the generated transition table looks different from the hand-

Table 1. Selected $k = 8$ cluster characterization on IW data. The Size column counts ground-truth segments assigned to each cluster, not primitive actions; the sizes sum to the 8,290 IW ground-truth segments. The action-share column lists nonzero within-cluster action percentages, and the trajectory column lists unique action-type sequences sorted by frequency. Clustering quality across different values of k is moved to Appendix Table A7.

Cluster	Size	Action Shares	Top Action Trajectories	Dominant GT Skill	Purity
C0	264	click (69%); scroll (27%); right-click (4%)	1. click -> scroll -> click -> click (112); 2. click -> scroll -> click -> click -> scroll (61); 3. click -> click -> click -> click -> scroll (46)	`monitor_status`	0.66
C1	2534	click (64%); type (36%)	1. click -> type -> type -> click (708); 2. click -> click -> type -> type -> click -> click (632); 3. click -> type -> click (632)	`send_message`	0.28
C2	2803	click (19%); select (19%); type (19%); format (19%); save (19%); scroll (6%)	1. click -> select -> type -> format -> save (1963); 2. click -> select -> type -> format -> save -> scroll (840)	`document_edit`	**1.00**
C3	110	click (38%); copy (19%); switch (19%); paste (19%); scroll (6%)	1. click -> copy -> switch -> click -> paste (74); 2. click -> copy -> switch -> click -> paste -> scroll (36)	`data_transfer`	**1.00**
C4	111	click (75%); right-click (25%)	1. click -> right-click -> click -> click (111)	`organize_files`	**1.00**
C5	71	click (100%)	1. click -> click -> click -> click (71)	`export_publish`	**1.00**
C6	2176	click (54%); type (24%); scroll (22%); save (1%)	1. click -> scroll -> click -> type -> click (694); 2. click -> scroll -> click -> type -> click -> scroll (312); 3. click -> type -> click -> scroll (300)	`review_content`	0.46
C7	221	click (60%); type (20%); save (20%)	1. click -> click -> type -> click -> save (221)	`presentation_edit`	**1.00**

Table 2. Completed model comparison for composing auto-generated skills. IW is the source benchmark; WebArena and BrowseComp+ are the held-out transfer checks used for the policy-transfer claim. Scores for these three columns are single-run skill-step percentages (%). WorkArena-NLP is an auxiliary text-only diagnostic and reports field accuracy on 100 structured-planning examples; it is not a live WorkArena substitute. Mind2Web is an auxiliary web-action diagnostic and reports task completion rate on 200 `test_task` examples. Cells report mean $\pm$ approximate binomial standard error in percentage points, computed from the single-run counts. API models are zero-shot format-constrained prompting baselines. The DeepSeek BrowseComp+ score is a verified near-zero result: 1/2,194 correct samples, or 0.0456%, rounded to 0.05. A dash marks a pending run. Best completed per-column scores are shown in **bold**.

Model	IW	WebArena	BrowseComp+	WorkArena-NLP	Mind2Web
Qwen3-8B (zero-shot)	18.5±1.9	55.8±1.7	43.5±1.1	37.0±4.8	9.5±2.1
Qwen3-8B (GRPO from base)	20.5±2.0	44.2±1.7	43.3±1.1	37.0±4.8	9.0±2.0
Llama-3.1-70B	**30.0±2.3**	56.2±1.7	51.9±1.1	38.0±4.9	9.5±2.1
OLMo-3-7B	14.3±1.8	54.5±1.7	**61.4±1.0**	12.8±3.3	34.0±3.3
Gemma-4-31B	16.5±1.9	37.7±1.7	11.2±0.7	38.6±4.9	–
DeepSeek-R1-Distill-32B	14.3±1.8	45.1±1.7	0.05±0.0	31.2±4.6	**36.0±3.4**
GPT-5	24.5±2.2	**57.6±1.7**	59.5±1.0	**40.6±4.9**	0.0±0.0
Claude Sonnet 4.5	25.8±2.2	**57.6±1.7**	47.7±1.1	39.8±4.9	15.0±2.5
Claude Haiku 4.5	27.8±2.2	50.8±1.7	51.0±1.1	39.6±4.9	20.0±2.8

written one, but whether it improves next-skill prediction. Relative to the hand-written table, the result is mixed: Auto-generated `SKILL.md` is better at $N = 100$, 250, and 2,000, but worse at $N = 500$ and 1,000.

The negative finding is sharper when the trivial baseline is included. Frequency has lower normalized edit distance at every evaluated training size; at $N = 2{,}000$, Frequency reaches 0.485 while Auto-`SKILL.md` reaches 0.528. Thus automated generation can beat a simple hand-written table in some settings, but it does not yet beat the strongest trivial baseline.

Performance depends on workflow complexity. Auto-generated files beat the hand-written table on high-complexity workflows (0.580 vs. 0.609) and medium-complexity workflows (0.517 vs. 0.560). Hand-written files do better on low-complexity workflows (0.506 vs. 0.552), where simple rules are enough.

We originally visualized the hand-written and auto-generated transition matrices as side-by-side heatmaps. That comparison was hard to interpret because the two matrices answer a low-level structural question: where does probability mass go after each skill? The more important evaluation question is downstream: does the generated specification improve next-skill prediction? We therefore report the data-efficiency curve in Figure 2, which compares methods on normalized edit distance, exact sequence match, and mean position accuracy.

7. Discussion

Readable mined structure is not the same as a repaired agent. Equation 2 finds many true boundaries but also over-splits within skills; Equations 3–5 produce readable clusters but remove action order; and the Phase 3 policies do not convert those labels into strong cross-domain behavior.

Mined-skill methods should be evaluated against frequency and transition priors before being compared to large language models. In our experiments, the most-common-skill baseline captures class imbalance and repetitive workflow structure that learned systems can appear to exploit. Future work should report a frequency prior, a transition-memory prior, and a modality-matched supervised policy.

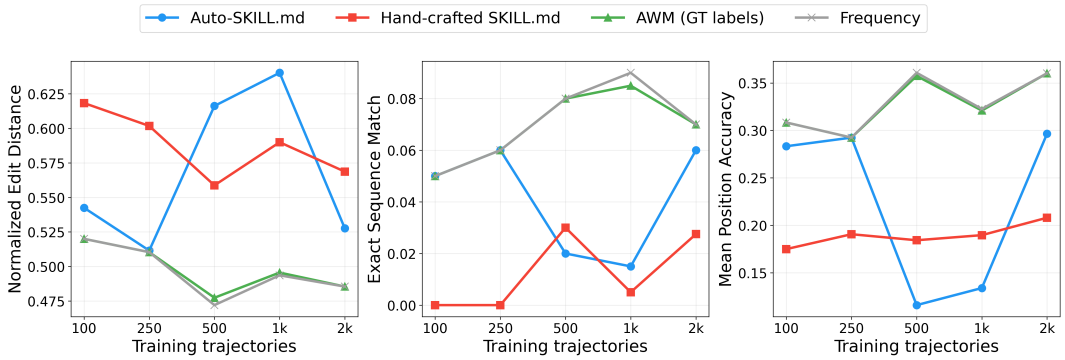

Figure 2. Data-efficiency comparison for generated `SKILL.md`. Lower normalized edit distance is better; higher exact match and mean position accuracy are better. The generated specification improves over the hand-written baseline at some sizes, including the largest setting, but it remains worse than the Frequency baseline across the evaluated sizes. This supports a cautious conclusion: trajectory-mined specifications can be competitive with simple manual tables, but the current pipeline is not yet a reliable replacement for trivial statistical baselines or manual design.

The small Transformer beats the GRPO-trained Qwen3-8B policy on IW, but the comparison changes input modality and optimization objective: the Transformer consumes segment embeddings and is trained for sequence prediction, while Qwen3-8B consumes text prompts and is optimized through a learned reward model. The result shows readable transition priors, not reusable skill-composition structure beyond dataset imbalance.

The limitations suggest concrete next steps. The contrastive encoder uses cluster-derived pseudo-labels; IW is synthetic; and the Phase 1 ℓ_2 boundary heuristic should be compared with learned action-prediction-error segmenters. Stronger claims would require improved Mind2Web GRPO results and live WorkArena evaluations. Follow-up experiments should mix IW with target-domain prompts, use order-aware segment encoders, train denser action-level or task-success rewards, and run factorial ablations with ground-truth boundaries, oracle target-domain boundaries, or supervised controls. Generated `SKILL.md` artifacts should remain human-reviewable before deployment.

Impact Statement

This work studies how to mine inspectable skill specifications for computer-using agents from interaction trajectories. The intended positive impact is to make agent behavior easier to audit, debug, and adapt: explicit `SKILL.md` files can expose intermediate routines that developers and domain experts can inspect before deployment. Such structure may be useful for repetitive enterprise workflows, accessibility tools, and agent evaluation, especially when compared against opaque end-to-end policies.

The same capability also creates risks. Reusable GUI skills could lower the cost of automating unwanted actions such as spam, credential misuse, bulk account manipulation, or other policy-violating workflows. Mined skills can also encode dataset biases, brittle interface assumptions, and unsafe action patterns if they are generated from narrow or synthetic traces. Our results argue against deploying these artifacts without human review: the learned policy does not show reliable transfer, and simple priors can outperform the proposed learned components. Any release of generated skills should therefore be treated as a research artifact, paired with provenance, benchmark limitations, and manual approval before use in live systems.

8. Conclusion

We mined skill specifications from GUI trajectories and evaluated whether they help downstream policies. The mined skills are readable on the source data: five of eight clusters reach at least 0.95 purity. The harder test is intervention. GRPO reaches 20.5% IW skill-step accuracy versus 18.5% zero-shot, and 43.3% BrowseComp+ accuracy versus 43.5% zero-shot. A Frequency baseline beats the proposed MLP and GRPO policies on IW and beats Auto-`SKILL.md` on edit distance at all evaluated data sizes. Trace-mined abstractions can therefore be inspectable yet operationally weak. Mined skills are useful diagnostic artifacts, but not yet a reliable replacement for simple baselines, manual design, or a cross-domain policy.

References

Ajay, A., Kumar, A., Agrawal, P., Levine, S., and Nachum, O. OPAL: Offline primitive discovery for accelerating offline reinforcement learning. In *International Conference on Learning Representations*, 2021.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Bai, H., Zhou, Y., Cemri, M., Pan, J., Suhr, A., Levine, S., and Kumar, A. DigiRL: Training in-the-wild device-control agents with autonomous reinforcement learning. In *Advances in Neural Information Processing Systems*, 2024.

Chen, M., Li, Y., Yang, Y., Yu, S., Lin, B., and He, X. AutoManual: Constructing instruction manuals by LLM agents via interactive environmental learning. In *Advances in Neural Information Processing Systems*, 2024.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transportation distances. In *Advances in Neural Information Processing Systems*, pp. 2292–2300, 2013.

Deng, J., Wang, Z., Cai, S., Liu, A., and Liang, Y. Open-world skill discovery from unsegmented demonstration videos. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10708–10718, 2025.

Deng, X., Gu, Y., Zheng, B., Chen, S., Stevens, S., Wang, B., Sun, H., and Su, Y. Mind2Web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36:28091–28114, 2023.

Dowson, D. C. and Landau, B. V. The Fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 12(3):450–455, 1982.

Drouin, A., Gasse, M., Caccia, M., Laradji, I. H., Del Verme, M., Marty, T., Boisvert, L., Thakkar, M., Cappart, Q., Vazquez, D., Chapados, N., and Lacoste, A. WorkArena: How capable are web agents at solving common knowledge work tasks? In *International Conference on Machine Learning*, 2024.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. In *International Conference on Learning Representations*, 2019.

Eysenbach, B., Salakhutdinov, R., and Levine, S. The information geometry of unsupervised reinforcement learning. In *International Conference on Learning Representations*, 2022.

Frans, K., Ho, J., Chen, X., Abbeel, P., and Schulman, J. Meta learning shared hierarchies. In *International Conference on Learning Representations*, 2018.

Ghosh, D., Gupta, A., and Levine, S. Learning actionable representations with goal-conditioned policies. In *International Conference on Learning Representations*, 2019.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. In *International Conference on Learning Representations*, 2017.

Guo, Y., Yang, W., Yang, S., et al. OpAgent: Operator agent for web navigation. *arXiv preprint arXiv:2602.13559*, 2026.

Hao, Y., Liu, Z., Riter, B., and Kalantari, S. Advancing patient-centered shared decision-making with AI systems for older adult cancer patients. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–19, 2024. doi: 10.1145/3613904.3642353.

Hao, Y., Alhamoud, K., Jeong, H., Zhang, H., Puri, I., Torr, P., Schaekermann, M., Stern, A. D., and Ghassemi, M. MedPAIR: Measuring physicians and AI relevance alignment in medical question answering. *arXiv preprint arXiv:2505.24040*, 2025a.

Hao, Y., Holmes, J., Waddle, M. R., Davis, B. J., Yu, N. Y., Vickers, K., Preston, H., Margolin, D., Lockenhoff, C. E., Vashistha, A., Kalantari, S., Ghassemi, M., and Liu, W. Personalizing prostate cancer education for patients using an EHR-integrated LLM agent. *npj Digital Medicine*, 2025b.

Khosla, P., Teterwak, P., Wang, C., Sarna, A., Tian, Y., Isola, P., Maschinot, A., Liu, C., and Krishnan, D. Supervised contrastive learning. In *Advances in Neural Information Processing Systems*, 2020.

Koh, J. Y., Lo, R., Jang, L., Duvvur, V., Lim, M. C., Huang, P.-Y., Neubig, G., Zhou, S., Salakhutdinov, R., and Fried, D. VisualWebArena: Evaluating multimodal agents on realistic visual web tasks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016.

Laskin, M., Liu, H., Peng, X. B., Yarats, D., Rajeswaran, A., and Abbeel, P. Unsupervised reinforcement learning with contrastive intrinsic control. In *Advances in Neural Information Processing Systems*, 2022.

Li, S., Wang, R., Tang, M., and Zhang, C. Hierarchical reinforcement learning with advantage-based auxiliary rewards. In *Advances in Neural Information Processing Systems*, 2019.

Li, X., Gao, M., Zhang, Z., Yue, C., and Hu, H. Selection of LLM fine-tuning data based on orthogonal rules. *arXiv preprint arXiv:2410.04715*, 2024.

Li, X., Gao, M., Hao, Y., Li, T., Wan, G., Wang, Z., and Wang, Y. MedGUIDE: Benchmarking clinical decision-making in large language models. *arXiv preprint arXiv:2505.11613*, 2025a.

Li, X., Gao, M., Zhang, Z., Fan, J., and Li, W. Data-adaptive safety rules for training reward models. *arXiv preprint arXiv:2501.15453*, 2025b.

Liu, G., Zhao, P., Liu, L., Chen, Z., Chai, Y., Ren, S., Wang, H., He, S., and Meng, W. LearnAct: Few-shot mobile GUI agent with a unified demonstration benchmark. *arXiv preprint arXiv:2504.13805*, 2025.

Nachum, O., Gu, S., Lee, H., and Levine, S. Data-efficient hierarchical reinforcement learning. In *Advances in Neural Information Processing Systems*, 2018.

Peyré, G. and Cuturi, M. Computational optimal transport: With applications to data science. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019.

Qi, Z., Liu, X., Iong, I. L., Lai, H., Sun, X., Zhao, W., Yang, Y., Yang, X., Sun, J., Yao, S., Zhang, T., Xu, W., Tang, J., and Dong, Y. WebRL: Training LLM web agents via self-evolving online curriculum reinforcement learning. In *International Conference on Learning Representations*, 2025.

Riemer, M., Liu, M., and Tesauro, G. Learning abstract options. In *Advances in Neural Information Processing Systems*, 2018.

Sarch, G., Jang, L., Tarr, M. J., Cohen, W. W., Marino, K., and Fragkiadaki, K. VLM agents generate their own memories: Distilling experience into embodied programs of thought. In *Advances in Neural Information Processing Systems*, 2024.

Sharma, A., Gu, S., Levine, S., Kumar, V., and Hausman, K. Dynamics-aware unsupervised discovery of skills. In *International Conference on Learning Representations*, 2020.

Sun, Q., Cheng, K., Ding, Z., Jin, C., Wang, Y., Xu, F., Wu, Z., Jia, C., Chen, L., Liu, Z., Kao, B., Li, G., He, J., Qiao, Y., and Wu, Z. OS-Genesis: Automating GUI agent trajectory construction via reverse task synthesis. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics*, pp. 5555–5579, 2025.

Sutton, R. S., Precup, D., and Singh, S. Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Tian, Y., Chen, J., Zheng, L., Tao, M., Zeng, X., Yin, Z., Su, H., and Sun, X. Skills-coach: A self-evolving skill optimizer via training-free GRPO. *arXiv preprint arXiv:2604.27488*, 2026.

Truong, C., Oudre, L., and Vayatis, N. Selective review of offline change point detection methods. *Signal Processing*, 167:107299, 2020. doi: 10.1016/j.sigpro.2019.107299.

Vezhnevets, A. S., Osindero, S., Schaul, T., Heess, N., Jaderberg, M., Silver, D., and Kavukcuoglu, K. FeUdal networks for hierarchical reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 3540–3549. PMLR, 2017.

Wang, X., Wang, B., Lu, D., et al. OpenCUA: Open foundations for computer-use agents. *arXiv preprint arXiv:2508.09123*, 2025.

Wang, Z. Z., Mao, J., Fried, D., and Neubig, G. Agent workflow memory. *arXiv preprint arXiv:2409.07429*, 2024.

Xie, T., Zhang, D., Chen, J., Li, X., Zhao, S., Cao, R., Hua, T. J., Cheng, Z., Shin, D., Lei, F., Liu, Y., Xu, Y., Zhou, S., Savarese, S., Xiong, C., Zhong, V., and Yu, T. OSWorld: Benchmarking multimodal agents for open-ended tasks in real computer environments. In *Advances in Neural Information Processing Systems*, 2024.

Xu, Y., Lu, D., Shen, Z., Wang, J., Wang, Z., Mao, Y., Xiong, C., and Yu, T. AgentTrek: Agent trajectory synthesis via guiding replay with web tutorials. In *International Conference on Learning Representations*, 2025.

Yang, Y., Yang, Z., Dou, Z.-Y., et al. UltraCUA: A foundation model for computer use agents with hybrid action. *arXiv preprint arXiv:2510.17790*, 2025.

Yao, S., Chen, H., Yang, J., and Narasimhan, K. WebShop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757, 2022.

Zheng, B., Fatemi, M. Y., Jin, X., Wang, Z. Z., Gandhi, A., Song, Y., Gu, Y., Srinivasa, J., Liu, G., Neubig, G., and Su, Y. SkillWeaver: Web agents can self-improve by discovering and honing skills. *arXiv preprint arXiv:2504.07079*, 2025.

Zhou, S., Xu, F. F., Zhu, H., Zhou, X., Lo, R., Sridhar, A., Cheng, X., Bisk, Y., Fried, D., Alon, U., and Neubig, G. WebArena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, 2024.

Zhou, Y., Yang, Q., Lin, K., Bai, M., Zhou, X., Wang, Y.-X., Levine, S., and Li, E. Proposer-agent-evaluator (PAE): Autonomous skill discovery for foundation model internet agents. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025.

A. Appendix

A.1. Additional GRPO Training Sessions

We also run a scale-control GRPO session on Llama-3.1-70B-Instruct with quantized low-rank adaptation (QLoRA) adapters. This anti-collapse rerun starts from the base model, uses 2 candidate responses per prompt, maximum completion length 64, learning rate 10^{-5}, gradient accumulation 16, and one epoch. The reward is a hand-weighted skill-format reward with components `correct_skill(+1.0)`, `format(+0.1)`, `reasoning(+0.1)`, and `invalid(-0.3)`. The run completes 1,149 optimizer steps in 224,341 seconds on 8 NVIDIA RTX A6000 GPUs with 49,140 MiB memory per GPU, with final training loss -1.16×10^{-4}, mean reward 0.529, reward standard deviation 0.460, and mean entropy 0.277.

A.2. Hyperparameters

Table A1 summarizes the model and optimizer settings used for the encoder, sequence baseline, and GRPO runs.

Table A1. Complete hyperparameters for all model configurations.

Component	Hyperparameter	Value
Phase 2 Encoder	Architecture	MLP: $30 \to 64 \to 32 \to 16$
	Optimizer	AdamW, lr=10^{-3}, wd=10^{-4}
	Training	200 epochs, batch 256, $T = 0.07$
Phase 3 Transformer	Architecture	2-layer, 4 heads, $d = 64$
	Optimizer	Adam, lr=10^{-4}
	Training	500 epochs, teacher forcing
GRPO	Base model	Qwen3-8B
	Candidates per prompt	8
	Temperature	0.7
	Optimizer	cosine, lr=5×10^{-6}
	Training	1 epoch, 4 GPUs, 6,072 seconds
	Reward	trajectory reward model, clip=5.0
	Base model	Llama-3.1-70B-Instruct
	Candidates per prompt	2
	Maximum completion length	64
	Optimizer	cosine, lr=10^{-5}
	Training	1 epoch, 8 GPUs, 224,341 seconds
	Reward	hand-weighted skill-format reward

A.3. Assets, Licenses, and New Artifacts

We use third-party benchmarks and models only as cited research artifacts or through their public provider interfaces. The new dataset introduced by this work are the synthetic IW trajectories, mined cluster assignments, generated `SKILL.md` specifications, evaluation prompts/predictions derived from our runs, and the WorkArena-NLP diagnostic conversion.

Section 5 documents the dataset composition and curation logic; Appendix B documents benchmark status and diagnostics; Appendix D documents generated skill specifications and examples. Code and public-release artifacts are available through the anonymous project repository linked in the main text.

A.4. GRPO Reward Model

The current GRPO run uses a learned trajectory reward model rather than a hand-weighted format reward. The reward model is a Qwen3-8B transformer with a scalar sequence-classification head, loaded through `AutoModelForSequenceClassification(num_labels=1)`. For an input string x, it returns one real-valued logit $r_\phi(x)$. It is trained with the Transformer Reinforcement Learning (TRL) library's `RewardTrainer` on pairwise preferences over full skill plans. The input text is the same planning prompt used by the policy concatenated with a candidate completion of the form `[Plan] skill_1 -> skill_2 -> ...`.

The reward-model data comes from the train and validation JSONL files under `data/interaskill/conversations/`. These files contain 1,275 train conversations and 225 validation

conversations with task text, message history, and a ground-truth `skill_flow`. We keep examples with at least two skills. For each prompt, the ground-truth skill flow is the chosen completion. Rejected completions are synthetic hard negatives generated by adjacent swaps, single-skill replacements, deletions, insertions, global swaps, and duplicate/mutation operations. The training run uses three negatives per train prompt and two negatives per validation prompt, producing approximately 3,825 train preference pairs and 450 validation pairs.

Preference margins are generated heuristically rather than from held-out benchmark correctness. A candidate plan is scored against the ground-truth skill flow by a weighted combination of prefix match (0.45), longest-common-subsequence overlap (0.30), unordered skill overlap (0.20), and length agreement (0.05). The chosen plan receives the self-score; each rejected plan receives its heuristic score, and the preference margin m is clipped to $[0.05, 0.95]$. TRL's reward training objective is a pairwise logistic ranking loss over the chosen and rejected scalar logits, with the margin requiring the chosen score to exceed the rejected score by more than the heuristic gap:

$$\mathcal{L}_{\mathrm{RM}} = -\log \sigma\big(r_\phi(p, y^+) - r_\phi(p, y^-) - m\big)\,, \tag{6}$$

where p is the prompt, y^+ is the ground-truth skill-flow completion, and y^- is a synthetic near-miss plan. Thus the reward model learns to rank IW-style skill plans by similarity to the annotated skill flow. It is not trained on WebArena, WorkArena, BrowseComp+, Mind2Web, or live task-success labels.

There is no separate methodology for labeling "held-out benchmark correctness" for reward-model training: those labels are not used. Validation during reward-model training uses held-out IW conversations from the validation split in `data/interaskill/conversations/` with the same synthetic preference construction as training. Held-out benchmark correctness is measured only after GRPO by running the policy evaluator on the completed IW, WebArena, and BrowseComp+ checks. This distinction is important because the learned reward is a proxy for IW skill-flow similarity, not an estimator of cross-domain task success.

Reward-model training uses Qwen/Qwen3-8B, maximum length 768, one epoch, per-device batch size 1, gradient accumulation 32 in the reported script, learning rate 10^{-5}, weight decay 0.01, warmup ratio 0.1, cosine scheduling, bf16, and gradient checkpointing. The reported script can optionally load the reward backbone in 4-bit, but the final trajectory reward model used by GRPO is saved as a standard sequence-classification checkpoint. During GRPO, the reward function concatenates the policy prompt and completion, tokenizes to maximum length 1024, runs the reward model, and returns the scalar logit. The reported GRPO run clips this scalar to $[-5, 5]$ before within-group advantages are computed. This design avoids directly rewarding superficial format features, but it means reward quality is limited by the IW skill-flow preference construction and by the mismatch between text-plan rewards and downstream benchmark correctness.

B. Benchmark Status and Transfer Diagnostics

B.1. Detailed Experimental Setup

Datasets. The claimed pipeline evaluation uses one source dataset and two held-out transfer checks, with two additional diagnostics. IW is the source dataset because it provides ground-truth skill boundaries and labels: 2,000 fabricated interaction trajectories spanning 12 skill types (e.g., `document_edit`, `search_navigate`, `send_message`), totaling 8,290 ground-truth segments and 40,774 primitive actions. We curate IW to cover repeated enterprise routines such as document editing, communication, file organization, status monitoring, and cross-application data transfer. WebArena (Zhou et al., 2024) contributes 1,000 real map-navigation trajectories with 3,140 segments; it is useful as a stress test for realistic but homogeneous web behavior. BrowseComp+ contains trajectories derived from BrowseComp and tests complex browsing transfer with the same skill-aware response format as IW. Mind2Web (Deng et al., 2023) is used only as a web-action diagnostic in this submission. WorkArena (Drouin et al., 2024) is the closest enterprise transfer benchmark, but live WorkArena results are not claimed; WorkArena-NLP is a text-only diagnostic constructed by converting WorkArena task schemas into natural-language goals paired with structured JSON targets across navigation, sorting, filtering, record creation, and service-catalog ordering.

Models and baselines. For Phase 3 skill composition, we evaluate Qwen3-8B in zero-shot and GRPO-trained settings. We compare against zero-shot Llama-3.1-70B, DeepSeek-R1-Distill-Qwen-32B, Gemma-4-31B, OLMo-3-7B, GPT-5, Claude Sonnet 4.5, and Claude Haiku 4.5. We additionally include three non-LLM baselines: a manually authored `SKILL.md` transition table, AWM-style bigram transitions (Wang et al., 2024), and a most-frequent-skill predictor.

Table A2. Auto-generated vs. hand-crafted skill specifications. IW Acc. is InteraSkill Workflows skill-sequence accuracy and WebArena (WA) Acc. is WebArena skill-sequence accuracy. Mean position accuracy is reported. The hand-crafted baseline is a simple transition table, not an optimized expert system. The Frequency baseline is trivial but strong, outperforming the proposed MLP and GRPO policies on IW skill-step accuracy.

Model	Type	IW Acc.	WA Acc.	IW Edit Dist.
SKILL.md	Fixed table	0.140	0.087	0.633
Frequency	Most common	0.349	0.285	0.480
AWM (Wang et al., 2024)	Learned transitions	0.334	**0.788**	0.479
MLP (ours)	Embedding-based	0.233	0.285	0.557
Transformer (ours)	Sequence model	0.346	0.410	0.481
Qwen3-8B GRPO (ours)	Policy optimization	0.205	0.442	0.672

Table A3. Skill composition on IW data using mined skills. This table is not a controlled input-modality or reward-model ablation: the MLP and Transformer operate on continuous segment embeddings, while the GRPO-trained Qwen3-8B policy operates on text trajectory prompts and is optimized through a learned reward model. The rows should therefore be read as pipeline variants, not as evidence that the reward model alone causes the performance gap.

Model	Accuracy	Exact Match	Edit Dist.	Params	Training Data
MLP	0.233	0.054	0.557	5.6K	Embeddings
Transformer	0.346	0.079	0.481	45K	Embeddings
Qwen3-8B GRPO	0.205	0.000	0.672	8B	Trajectory prompts

Metrics. For Phase 1 segmentation we report boundary precision, recall, and F1 against ground-truth skill boundaries. For Phase 2 clustering we report Normalized Mutual Information (NMI), silhouette score, and purity, both on the raw Wasserstein clusters and on the supervised-contrastive refined latent space. For Phase 3 skill composition we report per-position accuracy, exact sequence match rate, and normalized edit distance between predicted and ground-truth skill sequences. Per-position accuracy reveals error compounding along the sequence; edit distance captures the cost of fixing a predicted sequence in tokens. For the Mind2Web diagnostic, a task is counted as complete under task completion rate (TCR) only when every predicted action matches the ground-truth action in sequence; we additionally report partial task success, skill accuracy, and per-step action accuracy.

B.2. Additional Transfer Tables

Tables A2–A5 provide supporting transfer results that are useful for context but are not the headline claims in the main text. They also expose a negative result: on IW, the most-common-skill Frequency baseline is stronger than the proposed MLP and GRPO policies, so these learned methods should not be interpreted as solving skill composition.

B.3. WorkArena Diagnostic Status

Live WorkArena is not part of the claimed evaluation in this submission. It would be a stronger test of enterprise transfer than WebArena because its tasks involve knowledge-base lookup, record editing, forms, lists, dashboards, and service catalog workflows, which are close to the workflow types that IW tries to simulate.

We added an offline WorkArena evaluation path for future use. It uses the same conversation format as IW, WebArena, and BrowseComp+. It expects WorkArena rollouts under `data/workarena/trajectories/`, converts them to JSONL conversations under `data/workarena/conversations/`, and evaluates skill prediction with `interaskill.eval_model`. Because the rollout file has not been collected, no live WorkArena result is reported or used in the paper's transfer claims.

We also construct a text-only WorkArena-NLP variant for diagnosing whether models understand the enterprise task specifications independently of browser control. On the first 100 examples, completed runs reach 12.8–40.6% field accuracy and 0% exact match, showing that partial schema recovery is possible but full structured reconstruction remains difficult. This is not a replacement for live WorkArena: it removes visual grounding, state tracking, clicking, and form interaction. It instead tests instruction understanding and workflow planning. Appendix B.4 gives the conversion details.

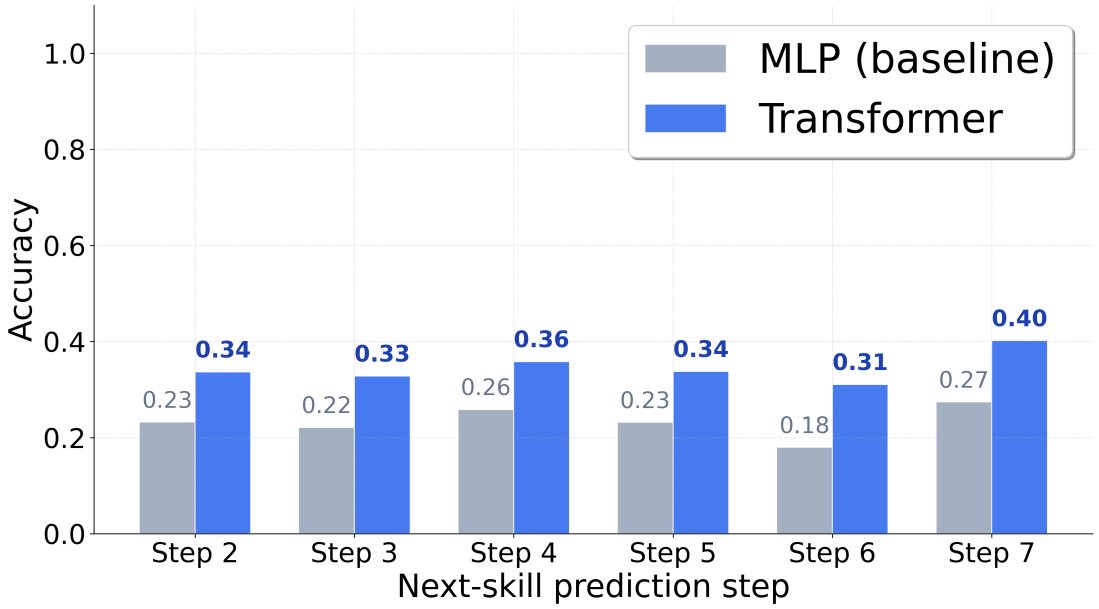

Figure A1. Next-skill prediction accuracy for MLP versus Transformer on IW data. Step 2 predicts the second skill given the ground-truth first skill; the seeded first skill at Step 1 is omitted. Both models operate on learned skill embeddings from Phase 2. Accuracy degrades at later steps due to error compounding, with the Transformer's sequential attention providing consistent improvement.

Table A4. Mind2Web diagnostic results on the `test_task` split. TCR is task completion rate; PTS is mean partial task success. These runs are reported as web-action diagnostics, not as evidence for the claimed IW-to-transfer skill-composition result.

Configuration	TCR	PTS	Skill Acc.	Action Acc.
Qwen3-8B zero-shot	9.5%	0.273	10.4%	28.8%
Qwen3-8B GRPO	9.0%	0.291	10.0%	30.8%
Llama-3.1-70B	9.5%	0.327	10.1%	31.6%
OLMo-3-7B	34.0%	0.815	22.3%	81.4%
DeepSeek-R1-Distill-32B	**36.0%**	**0.835**	**60.0%**	**83.4%**
GPT-5	0.0%	0.023	8.8%	2.1%
Claude Sonnet 4.5	15.0%	0.504	34.4%	46.7%
Claude Haiku 4.5	20.0%	0.586	22.8%	58.9%

B.4. Text-Only WorkArena Conversion

WorkArena is originally a live computer-using-agent benchmark. A model observes a ServiceNow page and must issue browser actions such as `click`, `fill`, `select_option`, `scroll`, and `press`. The ground truth is therefore an environment state, not a text label. For example, the basic menu-navigation task is correct only when the browser reaches the exact target ServiceNow URL; form and list tasks are correct only when the corresponding record, filter, sort order, or catalog order exists in the web application.

To separate task understanding from browser grounding, we define a derived text-only benchmark, WorkArena-NLP. We do not use screenshots, accessibility trees, or live browser state. Instead, we read WorkArena's configuration files and task schemas and convert each task into a pair (g, y), where g is the natural-language goal shown to the agent and y is a structured JSON plan containing the fields that the live task evaluator would eventually check. The model is prompted with g and must output y.

The conversion preserves the semantic target of each task family:

- Navigation tasks from `all_menu.json` become structured plans with `task_type`, `application`, `module`, `target_url`, and `required_final_action`.
- Sorting and filtering tasks expose the list name, filter kind, filter fields, filter values, sort fields, and sort directions.

Table A5. Per-domain Mind2Web results for Qwen3-8B zero-shot (test_task split).

Domain	n	**TCR (zero-shot)**
Entertainment	57	8.8%
Shopping	63	19.0%
Travel	80	2.5%

- Record-creation tasks expose the ServiceNow table, requested fields, field labels, target values, and `required_final_action=submit_record`.
- Service-catalog orders expose the item, description, quantity, item configuration values, and `required_final_action=order_item`.

The generated dataset contains 2,210 examples: 1,000 navigation examples, 300 sort examples, 300 filter examples, 250 create-record examples, and 360 catalog-order examples. We score model outputs by parsing the returned JSON and comparing it with the expected plan. Exact match requires all flattened fields to match after simple string normalization. Field accuracy is the fraction of expected flattened fields that match. This metric is intentionally stricter than semantic similarity but easier to audit.

WorkArena-NLP should be interpreted as a diagnostic benchmark. It answers whether a model can parse an enterprise workflow instruction into the right structured intent. It does not evaluate visual grounding, dynamic state tracking, login/session handling, menu expansion, clicking, form entry, or recovery from web-interface errors. Thus, WorkArena-NLP results are complementary to live WorkArena results rather than substitutes for them.

B.5. Mind2Web Diagnostic Status

Mind2Web is a stronger test of cross-domain web action transfer than the skill-sequence checks in Table 2, but it mixes skill choice with primitive action grounding. Table A4 therefore reports completed Mind2Web runs as diagnostics, not as evidence for the proposed policy.

Base Qwen3-8B reaches 9.5% task completion, 0.273 partial task success, 10.4% skill accuracy, and 28.8% action accuracy on Mind2Web `test_task`. The GRPO run reaches 9.0% task completion, so it does not beat the zero-shot baseline. Zero-shot performance varies by domain: Shopping is easiest and Travel is hardest (Table A5). The present submission makes no Mind2Web or live WorkArena transfer claim.

These diagnostics do not demonstrate that the IW skill vocabulary transfers to Mind2Web. They only show that the source-domain clusters are readable and that the current policy result is weak on the benchmarks we verified.

C. Skill Discovery Diagnostics

C.1. Segmentation Threshold Sensitivity

Table A6 reports the source-domain sweep used to select the action-discontinuity threshold θ.

Table A6. Sensitivity of segmentation F1 to the percentile used for θ on IW. Stable within a ± 10-percentile window around the optimum.

Percentile	θ	F1
40	1.41	0.527
45	1.49	0.534
50	1.545	**0.538**
55	1.60	0.535
60	1.66	0.524

C.2. Embedding Visualization

Figure A3 visualizes the learned segment embedding space used to inspect whether pseudo-label refinement separates source-domain skill types.

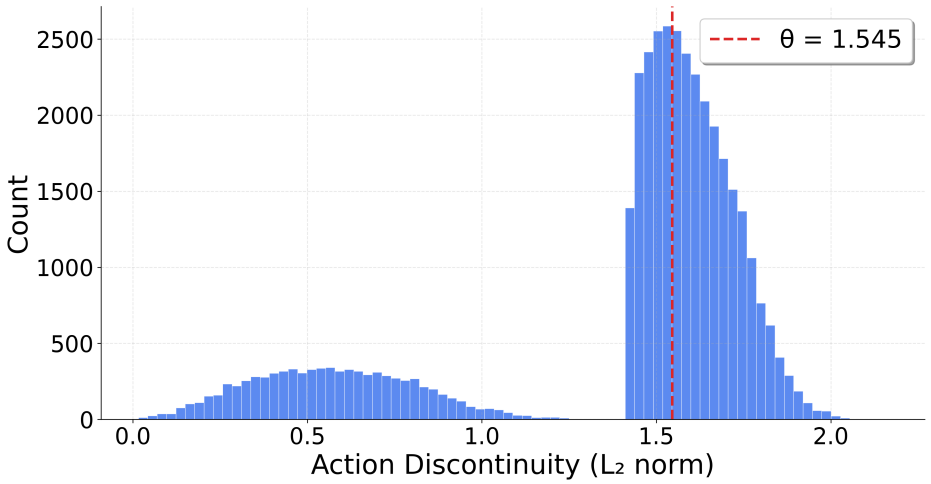

Figure A2. IW action-discontinuity scores. The dashed line marks the selected source-domain threshold $\theta = 1.545$.

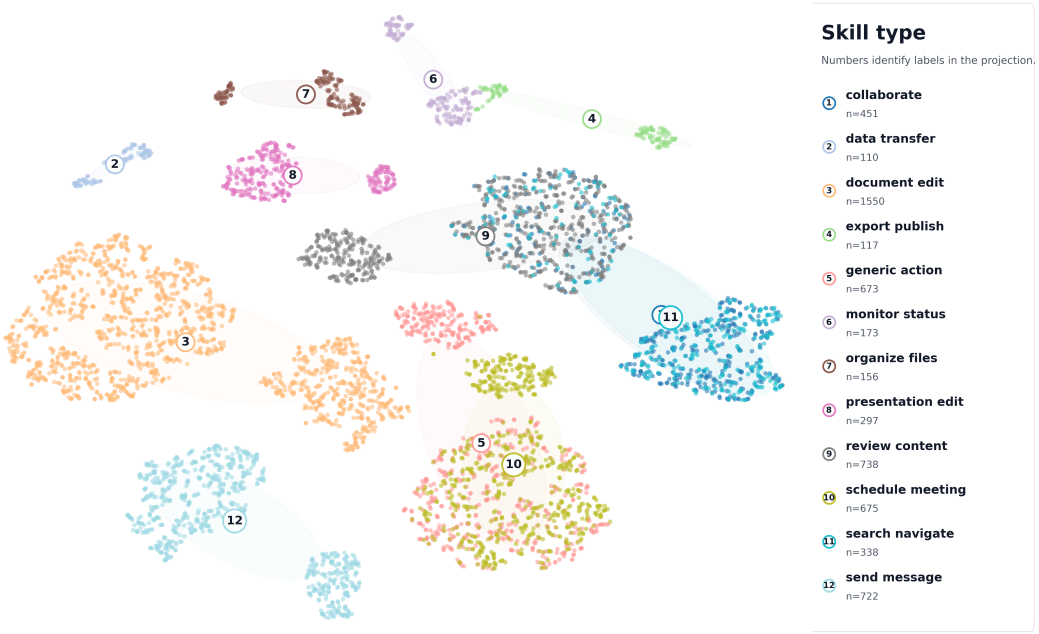

Figure A3. t-distributed stochastic neighbor embedding (t-SNE) projection of 16-dim skill embeddings on the IW benchmark. Colors and numbered badges indicate ground-truth skill types, with the embedded legend mapping each badge to a skill name. Well-separated groups confirm discriminative representation learning.

C.3. Wasserstein Cluster Count Sweep

Table A7 reports the unsupervised cluster-count sweep used to choose the $k = 8$ clustering analyzed in the main text.

C.4. Action Distribution Diagnostic

Figure A4 gives the action-type distribution behind the qualitative cluster descriptions in Table 1.

C.5. Contrastive Training Curves

Figure A5 shows the supervised-contrastive optimization trajectory for the skill encoder.

Table A7. Wasserstein clustering quality across different numbers of clusters k on IW. We use $k = 8$ in the main analysis because it gives the best NMI under the corrected diagonal 2-Wasserstein metric.

k	NMI	Purity
8	**0.650**	0.628
10	0.621	0.633
12	0.564	0.633
14	0.563	0.633
16	0.560	0.633

Figure A4. Per-cluster action-type distributions for the 8 auto-discovered clusters on IW data. Each row is a cluster, labelled with its dominant ground-truth skill and purity; columns are action types.

C.6. Per-Skill Accuracy and Confusion Matrix

Table A8 reports skill-level accuracy and Figure A6 shows how discovered clusters align with ground-truth skill labels. Here accuracy is the per-step next-skill, or tool-call, classification accuracy: among held-out IW steps whose ground-truth next skill is a given row label, it is the fraction for which the model predicts that same skill label. It is not primitive UI-action accuracy and not exact sequence match.

C.7. Qualitative Skill Descriptions

For each of the 8 discovered clusters in Table 1, we asked Claude Haiku 4.5 to describe the cluster given its top action types, dominant ground-truth skill label, purity, and three nearest-to-centroid exemplar segments. Thus these descriptions are not a blind semantic validation of the clusters: the LLM saw the plurality ground-truth label used in the table. We use them only as concise natural-language interpretations of the same cluster statistics and exemplars reported in Table 1. High-purity clusters correspond to cleaner behavioral patterns; low-purity clusters should be read as mixed action motifs rather than clean skills.

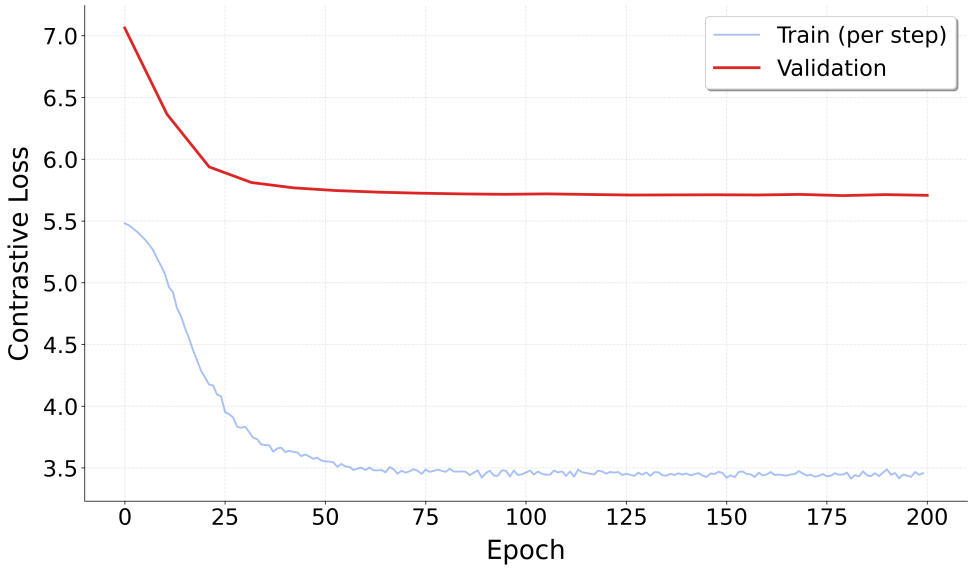

Figure A5. Supervised-contrastive training and validation loss over 200 epochs. The horizontal axis denotes epoch. Smooth convergence with stable validation loss indicates no overfitting.

Table A8. Per-skill next-skill/tool-call prediction accuracy on IW data, sorted by accuracy. Std. is the binomial standard error over per-step correctness for the single held-out evaluation, and 95% CI is the Wilson interval.

Skill	**Acc.**	**Std.**	**95% CI**	n
search_navigate	0.941	0.033	[0.841, 0.980]	51
organize_files	0.917	0.080	[0.646, 0.985]	12
data_transfer	0.897	0.049	[0.764, 0.959]	39
review_content	0.893	0.058	[0.728, 0.963]	28
document_edit	0.878	0.038	[0.785, 0.935]	74
collaborate	0.831	0.044	[0.727, 0.901]	71
export_publish	0.821	0.072	[0.644, 0.921]	28
send_message	0.809	0.048	[0.700, 0.885]	68
schedule_meeting	0.778	0.098	[0.548, 0.910]	18
presentation_edit	0.556	0.166	[0.267, 0.811]	9

- C0 (`monitor_status`, purity 0.66): Users inspect status-oriented interfaces through repeated click and scroll actions, often drilling into details and returning to nearby controls. The cluster is interpretable as monitoring behavior, but its moderate purity indicates overlap with other navigation-heavy skills.
- C1 (`send_message`, purity 0.28): Users perform broad click/type routines such as focusing a text field, entering content, and clicking again to submit or advance. Although the plurality label is `send_message`, the low purity shows that this cluster mixes several text-entry skills.
- C2 (`document_edit`, purity 1.00): Users perform document editing and formatting tasks by clicking to position the cursor, selecting and modifying text, applying formatting, and saving changes.
- C3 (`data_transfer`, purity 1.00): Users copy information from one application and paste it into another by selecting content, copying, switching applications, positioning the cursor, and inserting the data.
- C4 (`organize_files`, purity 1.00): Users organize files or folders by selecting an item, opening a context menu, and executing an organizational action.
- C5 (`export_publish`, purity 1.00): Users complete an export or publish workflow through a short sequence of confirmation clicks, consistent with finalizing an already configured artifact.
- C6 (`review_content`, purity 0.46): Users review or annotate displayed content by scrolling, selecting interface items, entering text, and moving through follow-up clicks. The cluster captures a reusable review-like action motif, but

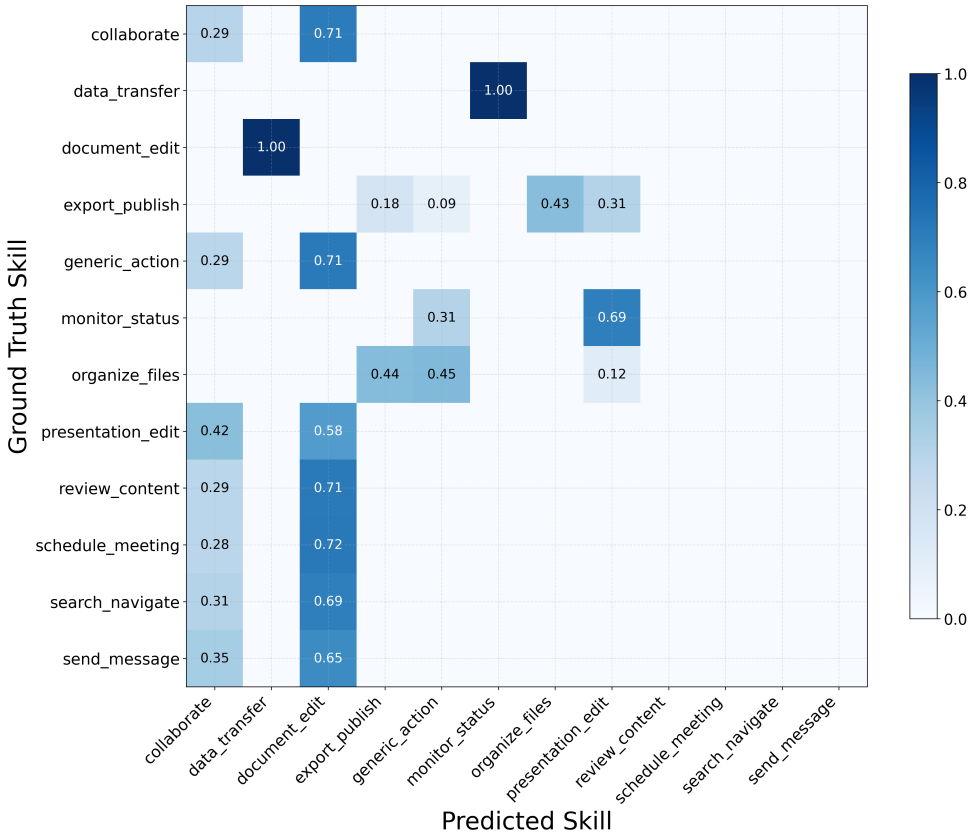

Figure A6. Normalized alignment between Wasserstein clustering assignments and ground-truth skill labels on IW data. Rows are discovered clusters and columns are ground-truth skill labels.

its sub-0.5 purity means it is not a clean one-skill cluster.

- C7 (`presentation_edit`, purity 1.00): Users edit presentation content by locating content, typing updates, and saving.

D. Generated Skill Specifications

This section supports the automated `SKILL.md` analysis in Section 6.5. The table gives the data-efficiency numbers behind Figure 2; the examples show what the generated skill descriptions look like against hand-written specifications.

D.1. Data-Efficiency Table

Table A9 gives the numeric data-efficiency results behind the main-text generated-`SKILL.md` comparison.

D.2. Hand-Crafted vs. Auto-Generated Skill Examples

The following examples compare hand-crafted and auto-generated `SKILL.md` descriptions on realistic workflow tasks. They are qualitative examples, not additional metrics. The hand-crafted examples show the kind of detailed manual specification an engineer might write; the auto-generated examples instantiate trajectory-derived reusable structure with the concrete entities, fields, and goals from the task. Across the examples, the main difference is that hand-crafted skills include more expert validation and recovery rules, while auto-generated skills combine reusable action patterns with task-specific context.

Example 1: Service Ticket Update. Task: In an IT service-management portal, find an open incident by ticket number, update the assignment group and priority, add a work note, and save the record.

Table A9. Automated vs. hand-crafted `SKILL.md` generation on IW trajectories. Normalized edit distance is lower better. The hand-crafted baseline is a simple expert-authored transition table. Auto-`SKILL.md` improves over the hand-crafted table at some sizes but is worse than the Frequency baseline at every evaluated size.

Method	$N = 100$	$N = 250$	$N = 500$	$N = 1{,}000$	$N = 2{,}000$
Frequency	0.520	0.510	**0.472**	**0.494**	0.485
SKILL.md (hand-crafted)	0.618	0.602	0.559	0.590	0.569
AWM (ground-truth labels)	0.520	**0.510**	0.477	0.496	**0.485**
Auto-SKILL.md (ours)	**0.542**	0.512	0.616	0.640	0.528

Hand-crafted `SKILL.md`
`update_service_ticket`

- **When:** ticket, case, incident, or change request updates.
- **Preconditions:** logged in; search or list filter visible; ticket identifier known.
- **Procedure:** search exact ticket ID; open the matching record; verify record number; locate `Assignment group`, `Priority`, `State`, and `Work notes`; update requested fields only; save.
- **Validation:** record number unchanged; edited values visible; note appears in activity stream.
- **Recovery:** filter duplicate search results; stop on read-only fields; repair required-field errors before one retry.

Auto-generated `SKILL.md`
Cluster C0 mapped to `record_field_update`

- **Observed pattern:** click search/list field → type ID → open result → edit labeled fields → click save/update.
- **Task context:** target is an open incident; required fields are assignment group, priority, and work note.
- **Reusable steps:** locate the incident by ticket number; bind task values to nearby labels; enter only requested updates; commit changes.
- **Generated validation:** prefer exact record match; confirm edited labels still show requested values; final save/update terminates the skill.
- **Generated caution:** preserve fields not mentioned in the task.

Example 2: Spreadsheet-to-CRM Data Transfer. Task: Copy a customer's renewal amount and renewal date from a spreadsheet, switch to a CRM opportunity page, paste the values into the matching fields, and submit the update.

Hand-crafted `SKILL.md`
`transfer_spreadsheet_value_to_crm`

- **When:** move values from a spreadsheet, table, email, or report into a business system.
- **Preconditions:** source and destination are reachable; source row is identifiable; destination field names are known or inferable.
- **Procedure:** locate source row; select cell; copy; switch to CRM; verify account or opportunity; paste into matching field; repeat for each value; save.
- **Validation:** compare currency symbols, decimals, dates, and units; confirm destination record saved.
- **Recovery:** recopy on clipboard failure; accept destination reformatting only if semantically equivalent; disambiguate duplicate customer rows by renewal period.

Auto-generated `SKILL.md`
Cluster C6 mapped to `data_transfer`

- **Observed pattern:** click source cell → copy value → switch app/tab → click destination input → paste → confirm.
- **Task context:** source is the customer's spreadsheet row; values are renewal amount and renewal date; destination is the CRM opportunity page.
- **Reusable steps:** identify row by customer; copy each required value; switch to CRM; bind destination by field label; paste amount and date; submit update.
- **Generated validation:** compare visible pasted strings with source values; allow formatting normalization only after checking semantic equivalence.
- **Generated caution:** app switching is expected and should not reset the task state.

Example 3: Shared Drive File Organization. Task: In a shared drive, find the latest quarterly budget file, rename it with the approved naming convention, move it into the finance archive folder, and verify that the old copy is no longer in the working folder.

Hand-crafted `SKILL.md`
`organize_shared_drive_file`

- **When:** rename, move, archive, duplicate, or clean up files and folders.
- **Preconditions:** file browser open; source folder accessible; metadata sufficient to identify target file.
- **Procedure:** search or sort folder; identify latest matching file by name, date, and extension; select file; open context menu; rename; confirm; choose move; navigate to archive; confirm move.
- **Validation:** find new filename in destination; check extension and date; verify source folder no longer contains the moved file.
- **Recovery:** disambiguate equally recent files; avoid overwriting file-name conflicts; fall back from drag-and-drop to context-menu move.

Auto-generated `SKILL.md`
Cluster C7 mapped to `organize_files`

- **Observed pattern:** click file-like item → open actions menu → choose rename/move operation → click destination or confirmation.
- **Task context:** target is the latest quarterly budget file; destination is the finance archive folder; operation is rename then move.
- **Reusable steps:** locate latest matching file; select it; rename with requested convention; open move action; choose archive folder; confirm.
- **Generated validation:** confirm the renamed file appears in the destination and no longer appears in the working folder.
- **Generated caution:** menu labels override positional assumptions; confirmation clicks often terminate the skill.