SYNTHESIZING OCCUPATIONAL TASKS AT SCALE FOR PRODUCTIVITY AGENTS

Anonymous authors
Paper under double-blind review

ABSTRACT

Synthesizing training data for productivity agents requires more than generating task instructions. The tasks must reflect the work people actually perform in their occupations, and their input materials must provide the information needed to meet the task requirements. We introduce SynthWorker to synthesize occupational tasks and their input materials at scale. SynthWorker leverages news articles relevant to each occupation and sector to provide work context for task generation. Within this context, we use a task hierarchy as the unit of synthesis, constructing shared input materials through web retrieval and synthesis and refining task instructions based on their contents. During execution, intermediate deliverables become additional input materials for dependent tasks, reflecting how real work builds on earlier results. Sharing the materials reduces average input and output token use per task by $6.7\times$ and $9.5\times$ compared with constructing materials separately for each task. We synthesize 100K tasks across 44 occupations in nine sectors and use a teacher model to solve them in dependency order. Fine-tuning the 4B and 9B Qwen3.5 base models on the resulting trajectories yields SynthWorker-4B and SynthWorker-9B, improving GDPval win rates by 39.9 and 44.0 percentage points over their respective base models. These gains extend to the subset of APEX-Agents requiring file deliverables, where mean scores increase by 17.6 and 20.9 percentage points.

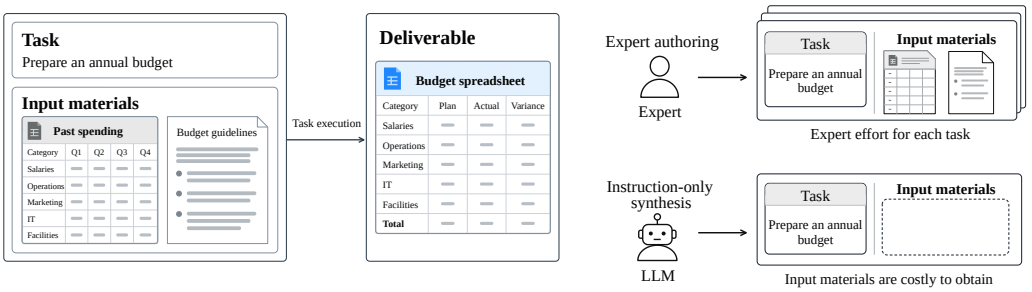
1 INTRODUCTION

Human societies create economic value through the production of goods and services. Across economic sectors, workers in different occupations perform tasks that contribute to this production. Productivity agents aim to carry out these occupational tasks using digital tools. Many of these tasks require input materials that provide information from a specific work context. For example, the annual budgeting task in Figure 1(a) uses past spending records and budget guidelines as input materials and requires a completed budget spreadsheet as its deliverable. Training agents for such work requires data that pair task instructions with the input materials needed to carry them out.

Benchmarks such as GDPval (Patwardhan et al., 2026) and APEX-Agents (Vidgen et al., 2026) evaluate productivity agents on realistic occupational tasks. Their construction relies on practitioners to write instructions and prepare input materials, making this approach costly to scale for training. Instruction synthesis (Wang et al., 2023; Xu et al., 2024; Wei et al., 2024) offers a way to expand training data, but Figure 1(b) illustrates why generating occupational task data requires more than task descriptions. The tasks must reflect the work people actually perform in their occupations, and their input materials must provide the information needed to meet the task requirements. The challenge is to synthesize tasks that reflect real occupational work, together with their input materials, at a cost that makes it practical to produce enough data for training.

We introduce SynthWorker to synthesize occupational tasks and their input materials at scale, aiming to cover a broad range of tasks encountered in real work. Following prior work on data synthesis from corpora (Qin et al., 2025; Ge et al., 2025), we retrieve news articles relevant to the target occupation and sector. We generate occupational tasks within the work context provided by each news article, using a task hierarchy as the unit of synthesis. Inspired by hierarchical task networks (Erol et al., 1994), each hierarchy decomposes a broader task into finer tasks, with every node

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.



(a) Example of an occupational task (b) Task generation approaches

Figure 1: Occupational task data include instructions and input materials. (a) A budgeting task with its input materials and deliverable. (b) Experts prepare both instructions and materials, whereas instruction synthesis leaves the materials to be obtained separately (dashed box).

independently assignable. A directed acyclic graph (DAG) records dependencies among sibling tasks, specifying which tasks require earlier deliverables.

We construct a shared pool of input materials for each hierarchy through web retrieval and synthesis, then refine task instructions based on their contents. This design allows related tasks to draw on different parts of the same input materials, spreading construction costs across the hierarchy. During execution, intermediate deliverables become input materials for dependent tasks, reflecting how real work builds on earlier results. Reusing these deliverables saves the additional cost of synthesizing input materials beyond the shared pool.

We synthesize 100K tasks across the 44 occupations and nine sectors covered by GDPval (Patwardhan et al., 2026). We use Qwen3.6-27B as the teacher model to generate trajectories by solving the task hierarchies in dependency order. Fine-tuning the 4B and 9B Qwen3.5 base models on these trajectories yields SynthWorker-4B and SynthWorker-9B. Their GDPval win rates improve over the corresponding base models by 39.9 and 44.0 percentage points, respectively. These gains extend to the file-deliverable subset of APEX-Agents (Vidgen et al., 2026), where the mean fraction of criteria passed increases by 17.6 and 20.9 percentage points. SynthWorker-4B also outperforms the released Qwen3.5-9B instruction-tuned model on both benchmarks. Shared input materials further lower the cost of constructing these tasks, reducing average input and output token use per task by factors of 6.7 and 9.5 compared with task-by-task construction.

The contributions of this work are threefold.

- We introduce SynthWorker to synthesize occupational tasks together with their input materials at scale. It uses news relevant to each occupation and sector as work context, without requiring a seed dataset of occupational tasks.
- We use task hierarchies as the unit of synthesis, allowing related tasks to share input materials and pass intermediate deliverables to dependent tasks. Shared material construction reduces average input and output token use per task by factors of 6.7 and 9.5 compared with constructing materials separately for each task.
- We show that the synthesized data improve both 4B and 9B base models on GDPval, with gains extending to the subset of APEX-Agents requiring file deliverables. SynthWorker-4B outperforms the released Qwen3.5-9B instruction-tuned model on both benchmarks, and a 4B student trained with a second teacher family also shows substantial gains.

2 Related Work

2.1 Agents for professional work

GDPval (Patwardhan et al., 2026), Agents' Last Exam (Sun et al., 2026), and APEX-Agents (Vidgen et al., 2026) evaluate productivity agents on tasks authored by practitioners from different occupa-

tions. The Remote Labor Index (Mazeika et al., 2025) and UpBench (Yi et al., 2025) obtain tasks from paid freelance work, preserving the requirements and deliverables of actual assignments. The accompanying input materials provide agents with the information needed to carry out the work. Public work materials also support task construction in SpreadsheetBench (Ma et al., 2024), Finch (Dong et al., 2026), and LongDA (Li et al., 2026), which draw on spreadsheets, enterprise documents, and survey data. Beyond providing files, AppWorld (Trivedi et al., 2024) and TheAgentCompany (Xu et al., 2025) construct application environments in which agents can execute tasks and their results can be checked. Across all these benchmarks, evaluation concerns whether an agent can complete an assignment using the available materials and tools.

For agent training, software engineering benefits from a large pool of open-source repositories containing code, development histories, and tests. These resources support task construction with inputs that agents can modify and tests that help verify their solutions. SWE-bench (Jimenez et al., 2024) mines repository issues for evaluation, while SWE-Gym (Pan et al., 2025) prepares repository-based tasks and executable environments for training. R2E-Gym (Jain et al., 2025) and SWE-smith (Yang et al., 2025) further generate training tasks from commits and code, allowing a repository to support many tasks. Outside software engineering, Spreadsheet-RL (Chi et al., 2026) obtains Excel tasks and their input spreadsheets from forums, while Rao et al. (2024) composes Word formatting tasks from API operations. These office-task methods organize task construction within a particular application. In SynthWorker, we instead define tasks by the work of a target occupation, which can involve different applications and input materials. We generate task hierarchies together with shared input materials to train productivity agents across occupations.

2.2 Synthetic data generation

Self-Instruct (Wang et al., 2023) expands a seed set by generating new instruction-response pairs, while Evol-Instruct (Xu et al., 2024) rewrites existing instructions into harder variants. GLAN (Li et al., 2025) instead specifies the intended coverage through a taxonomy of human knowledge and generates instructions across the disciplines in it. Corpora provide another basis for generation by supplying concrete content. OSS-Instruct (Wei et al., 2024) uses code snippets from repositories to generate coding problems. SynthLLM (Qin et al., 2025) extracts concepts from corpus documents and combines them through a graph, while PersonaHub (Ge et al., 2025) derives personas from web text to guide data generation. These corpus-based approaches connect the scope and content of synthetic data to the sources that guide generation. In SynthWorker, we specify the occupations to cover and retrieve relevant news to provide context for their tasks.

Environment synthesis complements task generation by providing the state and tools through which an agent can perform the work. InfiniteWeb (Zhang et al., 2026) generates functional websites and task evaluators, while Agent World Model (Wang et al., 2026) and ScaleEnv (Tu et al., 2026) construct executable tool environments with verifiable tasks. These environments allow agents to learn through interaction and feedback from task execution. For productivity work, Synthetic Computers (Ge et al., 2026) first expands a persona into a user profile and a file system of interlinked documents. It then generates work goals conditioned on the profile and the computer's contents. In SynthWorker, we instead generate a task hierarchy from a selected occupation and a relevant news article before constructing its shared input materials. The hierarchy makes shared materials and deliverable dependencies explicit while retaining independently assignable occupational tasks.

3 SynthWorker

Given an occupation and a sector, SynthWorker constructs task hierarchies with task instructions, input materials, and expected deliverables specified for each node (Figure 2). We first describe their construction from relevant news, then explain how we collect trajectories for supervised fine-tuning by solving the tasks in dependency order.

3.1 Occupation-Conditioned Retrieval

We use news articles to generate occupational tasks based on real-world events. To find articles relevant to each occupation, we search our corpus of 171 million CC-News (Nagel, 2016) articles

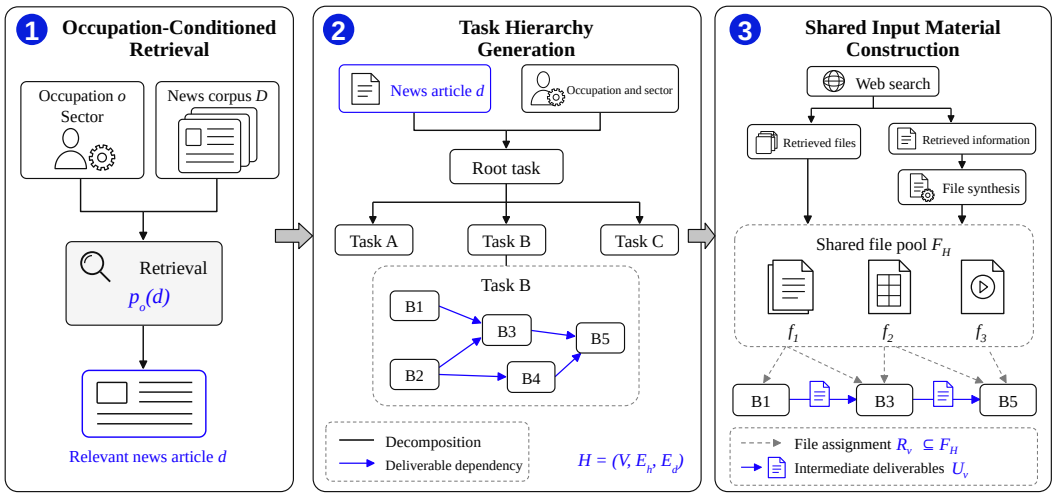


Figure 2: Overview of SynthWorker. Given an occupation and sector, we retrieve relevant news to generate task hierarchies with deliverable dependencies. We construct shared input materials through web retrieval and synthesis and assign them to tasks. During execution, intermediate deliverables become additional input materials for dependent tasks.

without occupation labels.[1] Labeling every article for each occupation with a language model is too expensive for our setting, as is encoding the full corpus for dense retrieval. We instead label a small set of articles and train a sparse classifier to find relevant articles across the full corpus on CPUs.

For each occupation and sector, we first retrieve candidate articles from a shared subset of one million CC-News articles encoded with Qwen3-Embedding-8B (Zhang et al., 2025). We use the query *"What is happening in the professional world of* `<occupation>` *in the sector of* `<sector>`*?"* to target news about the occupation's work and professional context rather than articles that merely mention the occupation. A language model then labels the top retrieved articles for occupational relevance. We retain articles labeled relevant as positive training examples and treat randomly sampled articles from the full corpus as negative examples.

Using these examples, we train a logistic regression classifier on TF-IDF features for each occupation. For article d and occupation o, the classifier assigns the relevance score

$$p_o(d) = \sigma\left(\mathbf{w}_o^\top \phi(d) + b_o\right). \tag{1}$$

Here, $\phi(d)$ is the article's TF-IDF feature vector, σ is the sigmoid function, and $\mathbf{w}_o$ and b_o are the parameters learned for occupation o. We apply each classifier to the full corpus, with articles satisfying $p_o(d) \geq 0.5$ forming the predicted positive pool for occupation o.

We then iteratively refine the classifier on the full corpus. In each round, we sample previously unlabeled articles from the classifier's predicted positive pool and label them with the same language model. These labels provide an estimate of precision, computed as the fraction of sampled articles labeled relevant. Articles labeled irrelevant serve as hard negatives for retraining. We retrain the classifier on all labels collected so far and rescore the full corpus, repeating the process until the classifier reaches target precision or the round limit. Finally, we randomly sample articles from the final positive pool and use a language model to confirm their relevance before task generation.

3.2 Task Hierarchies with Shared Input Materials

From each selected article, we generate a task hierarchy for the target occupation, inspired by hierarchical task networks (Erol et al., 1994; Georgievski & Aiello, 2015). The root specifies the overall task, which we decompose into progressively finer child tasks at the same or shorter time horizons, from months to hours of expected human work. Each node has its own instructions and expected deliverables and can be assigned independently. Deliverable dependencies specify which task outputs

[1] `https://huggingface.co/datasets/stanford-oval/ccnews`

become input materials for sibling tasks. These dependencies form a directed acyclic graph (DAG), distinct from the parent-child links. We write the hierarchy as

$$H = (V, E_{\mathrm{h}}, E_{\mathrm{d}}), \tag{2}$$

where V denotes the task nodes, E_{h} the parent-child relations, and E_{d} the deliverable dependencies.

In real work, input materials often contain more information than what a single task needs. Related tasks may draw on different parts of the same report, dataset, or other file. We therefore implement an agent to build shared input materials for the whole hierarchy. Let $\mathcal{F}_H$ denote the shared file pool and $R_v \subseteq \mathcal{F}_H$ the files assigned to task v. Building the shared pool once provides input materials for multiple independently assignable tasks, substantially reducing construction costs per task.

The agent first searches the web for the files these tasks need and uses them directly when available. If the agent cannot find a needed file, it creates the file from information gathered through the search. We also provide access to GPT-Image-2 for image synthesis and Sora-2 for video synthesis. Once the files are available, the agent assigns them to tasks and rewrites the task instructions to match their contents. It also specifies which deliverables each dependent task will receive.

3.3 Dependency-Ordered Execution

We collect trajectories for supervised fine-tuning by having a teacher model execute tasks at both parent and leaf nodes. Each task v runs in a fresh sandbox containing its assigned input materials R_v and the actual deliverables U_v produced by its predecessors. The inputs are

$$W_v = R_v \cup U_v. \tag{3}$$

We schedule these runs according to the deliverable-dependency DAG, launching each task only after its predecessors finish. Ready tasks run concurrently within and across hierarchies under this topological schedule. We record each run's messages and tool interactions as a trajectory, retaining it for fine-tuning only when the teacher finishes the task and all specified deliverables are present.

4 Experiments

4.1 Experimental Setup

We synthesize 100K tasks covering the same 44 occupations across nine sectors as GDPval (Patwardhan et al., 2026). Table 2 summarizes the task dataset. We use GPT-5.5 to label retrieved articles, generate task hierarchies, and build shared input materials. Using Qwen3.6-27B as the teacher, we collect training trajectories following Section 3.3. We fine-tune Qwen3.5-4B-Base and Qwen3.5-9B-Base on these trajectories for three epochs with a learning rate of 3×10^{-5} and a maximum sequence length of 131,072 tokens, yielding SynthWorker-4B and SynthWorker-9B. We compare them with the two base models, their instruction-tuned variants, and the teacher. Both teacher solving and benchmark evaluation use Stirrup (Artificial Analysis, 2025), the harness GDPval-AA (Artificial Analysis, 2026) uses. In the main evaluations, each agent runs in a fresh sandbox for up to 250 turns, with access to tools for code execution, web search, page fetching, and image viewing.

4.2 Benchmarks

GDPval. We evaluate 182 tasks from the open gold subset of GDPval (Patwardhan et al., 2026). Following the pairwise evaluation approach of GDPval-AA (Artificial Analysis, 2026), we use GPT-5.5 to compare each model's deliverables with the corresponding human deliverables. We repeat each comparison with the positions swapped, and one win for each side counts as a tie. We run each model three times per task and report the average win rate and win-or-tie rate.

APEX-Agents. We also evaluate the 58 file-deliverable tasks from APEX-Agents (Vidgen et al., 2026), covering the occupations of investment banking analyst, corporate lawyer, and management consultant. For these tasks, we use the benchmark's grader with GPT-5.5 to assess the final file-system state against task-specific rubrics. We report Pass@1, the fraction of tasks meeting all criteria, and mean score, the average fraction of rubric criteria met per task.

Table 1: Performance on GDPval and APEX-Agents. Qwen3.6-27B is the teacher model.

	GDPval (%)		APEX-Agents (%)	
Model	Win	Win or tie	Pass@1	Mean score
Qwen3.6-27B (teacher)	77.5	85.2	25.9	53.5
Qwen3.5-4B-Base	6.6	9.9	3.4	13.4
Qwen3.5-4B	20.9	26.2	0.0	18.1
Qwen3.5-9B-Base	14.3	17.8	1.7	15.8
Qwen3.5-9B	26.7	37.2	1.7	20.7
SynthWorker-4B	46.5 (+39.9)	57.0 (+47.1)	10.3 (+6.9)	31.0 (+17.6)
SynthWorker-9B	58.3 (+44.0)	64.5 (+46.7)	13.8 (+12.1)	36.7 (+20.9)

Table 2: SynthWorker dataset statistics. The final four rows report means from 100 task hierarchies in the construction-cost study. A material item is any file or folder a task references.

Statistic	Value
Tasks	100K
Occupations	44
Sectors	9
Mean tasks per hierarchy	25.40
Mean input material items per hierarchy	7.55
Mean input material items referenced per task	2.34
Mean tasks sharing each input material item	7.87

5 Results

5.1 Main Results

Fine-tuning improves both model sizes. As shown in Table 1, fine-tuning on trajectories from SynthWorker tasks increases GDPval win rate by 39.9 and 44.0 percentage points for the 4B and 9B base models, respectively. SynthWorker-4B and SynthWorker-9B reach win rates of 46.5% and 58.3% against human reference submissions on GDPval's expert-authored occupational tasks. Despite starting from a stronger base model, the 9B student gains slightly more than the 4B student. Both students also outperform their released instruction-tuned counterparts on all four metrics, with GDPval win-rate margins of 25.6 percentage points at 4B and 31.6 points at 9B. At 4B, this advantage extends to the larger Qwen3.5-9B instruction-tuned model. SynthWorker-4B outperforms it on all four metrics, leading by 19.8 points in GDPval win rate and 10.3 points in APEX mean score. It therefore achieves better results on both benchmarks with less than half as many parameters.

The gains transfer to APEX-Agents. Both students also improve on the subset of APEX-Agents requiring file deliverables. Here, the evaluation changes from comparison with human submissions to assessment against task-specific rubrics. Mean score improves by 17.6 percentage points at 4B and 20.9 points at 9B, reflecting a larger fraction of criteria met per task on average. The fraction of tasks meeting all criteria also increases, with Pass@1 rising from 3.4% to 10.3% at 4B and from 1.7% to 13.8% at 9B. GDPval win rate and APEX mean score rank all seven models identically despite the different tasks and grading protocols.

Both students narrow the gap to the 27B teacher. Fine-tuning reduces the GDPval win-rate gap to Qwen3.6-27B from 70.9 to 31.0 percentage points at 4B and from 63.2 to 19.2 points at 9B. On APEX-Agents, the mean-score gap falls from 40.1 to 22.5 percentage points at 4B and from 37.7 to 16.8 points at 9B. SynthWorker-9B reaches 75.3% of the teacher's GDPval win rate and 68.6% of its APEX mean score with one-third as many parameters.

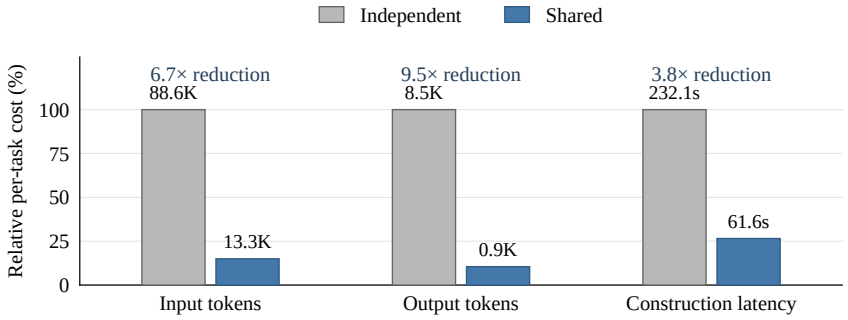


Figure 3: Per-task input material construction costs. Bar heights normalize to independent construction at 100%, and labels give absolute means. Shared costs, including construction latency, amortize over tasks within each hierarchy before averaging.

5.2 Efficient Input Material Construction

Sharing input materials substantially reduces construction cost per task. We sample one task from each of 100 completed hierarchies and rebuild its input materials independently with the same model and tools. We sum input and output tokens separately across the construction agent's calls. For shared construction, we divide each hierarchy's token totals and construction latency by its task count before averaging. As shown in Figure 3, sharing reduces input token use, output token use, and construction latency per task by factors of 6.7, 9.5, and 3.8, respectively.

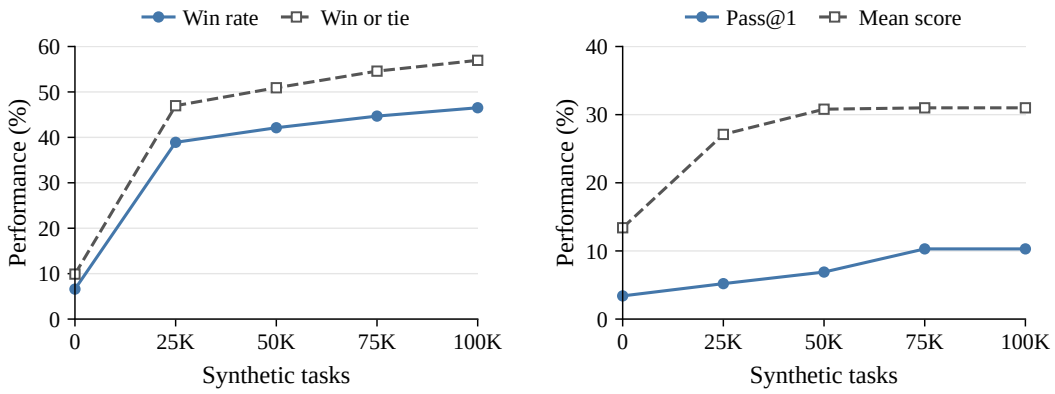
In the independent runs, tasks requiring supplied files consume 121.9K input tokens on average, compared with 14.5K for self-contained tasks whose instructions already contain the required information. Shared construction spreads the cost of supplying files across related tasks by reusing their input materials. To quantify this reuse, we count each referenced file or folder as one input material item. Tasks reference 2.34 input material items on average under shared construction, compared with 2.27 under independent construction. Each shared item serves 7.87 tasks on average. This is equivalent to preparing just 0.30 distinct material items per task on average, 87.3% fewer than if the same materials were prepared separately for each task.

5.3 Scaling with Synthetic Tasks

We compare Qwen3.5-4B-Base without fine-tuning with versions fine-tuned on 25K to 100K synthetic tasks, keeping the teacher and fine-tuning recipe fixed. As shown in Figure 4(a), moving from zero tasks to 25K raises GDPval win rate from 6.6% to 38.9% and win-or-tie rate from 9.9% to 47.0%. The first 25K tasks account for about 81% of the win-rate gain that 100K tasks deliver. With one quarter of the full training set, the 4B student already outperforms the released 9B instruction-tuned model in Table 1 on both GDPval metrics. We next examine how much performance improves as the training set grows beyond 25K tasks. Beyond 25K, successive additions of 25K tasks yield smaller win-rate gains of 3.2, 2.6, and 1.8 percentage points. Despite these diminishing returns, expanding from 25K to 100K still adds 7.6 points in win rate and 10.0 points in win-or-tie rate. On APEX-Agents, Figure 4(b) shows that moving from zero tasks to 25K raises mean score from 13.4% to 27.1% and Pass@1 from 3.4% to 5.2%. Expanding to 75K raises mean score to 31.0% and Pass@1 to 10.3%, with both unchanged at 100K. Most of the gain in mean score appears by 25K, whereas most of the gain in complete task success appears with further expansion. These results complement the construction savings in Section 5.2 by showing that additional synthesized tasks remain useful for training.

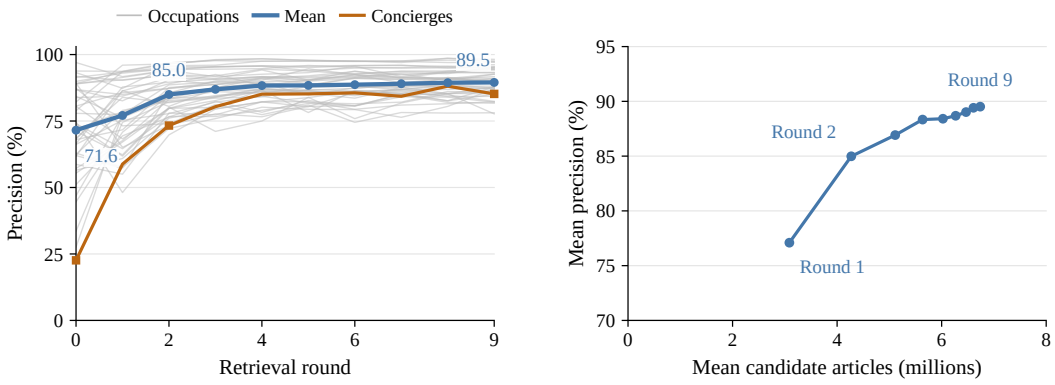
5.4 Improved Retrieval Precision at Scale

We extend retrieval from the one million articles encoded with Qwen3-Embedding-8B to the full 171-million-article corpus using a TF-IDF classifier on CPUs. As shown in Figure 5(a), mean precision across the 44 occupations increases from 71.6% under zero-shot retrieval to 89.5% in the final round. The first two updates bring precision to 85.0%, accounting for three quarters of the gain.



(a) GDPval (b) APEX-Agents

Figure 4: Performance on GDPval and APEX-Agents as the number of synthesized training tasks grows. Zero denotes Qwen3.5-4B-Base without fine-tuning.



(a) Precision by round (b) Precision and candidate pool size

Figure 5: Retrieval precision and candidate-pool size. (a) Precision for all 44 occupations, with the overall mean and the weakest zero-shot occupation, Concierges, highlighted. Round 0 is zero-shot retrieval. (b) Mean precision versus mean candidate-pool size for full-corpus rounds 1 through 9. Both coordinates use the same round's candidate pools.

Mean precision improves in every sector, as detailed in Appendix D.1. For the five occupations with zero-shot precision below 50%, mean precision rises from 35.1% to 86.4%. This includes Concierges, which starts with the weakest zero-shot result and improves from 22.6% to 85.2%.

These full-corpus precision estimates come from random samples of the predicted positive pools, rather than only the highest-ranked articles. As shown in Figure 5(b), the mean candidate pool grows from 3.1 million to 6.7 million articles between rounds 1 and 9, while mean precision rises from 77.1% to 89.5%. The updates thus increase both average precision and the supply of candidate articles for task generation. These results use the broad relevance criterion in Section 3.1, which accepts news about an occupation's work and field. A narrower query and matching relevance criterion could favor more specialized news at the cost of fewer candidates.

5.5 Robustness Across Teachers

Our main experiments use Qwen models as both teacher and student. To test whether the gains extend across teacher families, we fine-tune Qwen3.5-4B-Base on trajectories from 10K tasks solved

by either Qwen3.6-27B or GLM-5.2[2], keeping the fine-tuning recipe fixed. As shown in Table 5 in Appendix D.2, the two trajectory sets improve GDPval win rate by 21.4 and 27.3 percentage points over the base model, respectively. The gains also extend to APEX-Agents, where mean score reaches 26.5% and 27.1%, about twice the base model's 13.4%. Both students also increase Pass@1 from 3.4% to 5.2%, showing improvement in complete task success as well as average rubric scores. The gains with GLM-5.2 show that the training benefit is not confined to a teacher and student from the same model family.

5.6 Beyond Pairwise Evaluation

Pairwise grading is the main evaluation protocol in GDPval-AA (Artificial Analysis, 2026), and we use it to compare model deliverables with human reference submissions. To examine agreement across evaluation methods, we also score the deliverables using the expert-written rubrics that GDPval (Patwardhan et al., 2026) releases. Our grading agent inspects the files, checks each rubric criterion, and records evidence for its judgment. Writing these rubrics for new tasks is difficult to scale, and checking each criterion adds grading cost. Pairwise grading requires a reference submission but no human rubric.

To score deliverables without reference submissions or human rubrics, we examine market value grading. We use a grading agent to inspect the deliverable and estimate its value given a model-estimated task price. The market value score divides the deliverable value by the task price, capped at 100%. We use GPT-5.5 for all grading and run both grading agents in Stirrup. All three methods rank the five models in the same order. At the task level, market value scores correlate more closely with human-rubric scores than pairwise outcomes do, with mean Spearman correlations of 0.48 and 0.24 across the five models. Appendix D.3 provides the full results and scoring details. The market value score may be useful as a reward for rejection fine-tuning or reinforcement learning, but its effectiveness for training requires further validation.

6 Conclusion

With SynthWorker, we show that occupational training data can be synthesized without first collecting a dataset of workplace tasks and their input materials. SynthWorker leverages occupation-relevant news as context for generating task hierarchies and builds the input materials their tasks require. Building these materials once per hierarchy allows multiple tasks to share them, substantially reducing construction cost per task. The resulting tasks also support effective training, as fine-tuning on teacher-generated trajectories improves both 4B and 9B base models on GDPval. These results suggest that productivity-agent training need not be limited to work already represented in existing task datasets. Through careful construction of tasks and their input materials, we can build training environments designed around the demands of real-world work.

7 Limitations

Our experiments use a single agent harness, so transfer to other tool interfaces remains untested. We use a stronger teacher model to generate trajectories for supervised fine-tuning. Reducing reliance on the teacher model through rejection fine-tuning or reinforcement learning would require reliable feedback on task outcomes. Although SynthWorker constructs the required tasks and input materials, developing and validating such feedback at scale remains an open problem.

AI use statement

We used generative AI for synthetic data generation, code implementation, evaluation, and result analysis. We also used AI tools for literature search, manuscript drafting and polishing. The authors reviewed all AI-assisted work and take responsibility for the final content.

[2]The GLM-5.2 teacher is text-only, so we disable the image-viewing tool during trajectory generation. We retain it when evaluating the fine-tuned Qwen student. The comparison does not isolate teacher quality.

REFERENCES

Artificial Analysis. Stirrup: The lightweight framework for building agents, 2025. URL https://github.com/ArtificialAnalysis/Stirrup.

Artificial Analysis. Artificial analysis intelligence benchmarking methodology, 2026. URL https://artificialanalysis.ai/methodology/intelligence-benchmarking.

Banghao Chi, Yining Xie, Mingyuan Wu, Jingcheng Yang, Jize Jiang, Zhaoheng Li, Shengyi Qian, Minjia Zhang, Klara Nahrstedt, Rui Hou, Xiangjun Fan, and Hanchao Yu. Spreadsheet-RL: Advancing Large Language Model Agents on Realistic Spreadsheet Tasks via Reinforcement Learning, 2026. URL https://arxiv.org/abs/2605.22642.

Haoyu Dong, Pengkun Zhang, Yan Gao, Xuanyu Dong, Yilin Cheng, Mingzhe Lu, Adina Yakefu, and Shuxin Zheng. Finch: Benchmarking Finance & Accounting across Spreadsheet-Centric Enterprise Workflows. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 10768–10794, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.523. URL https://aclanthology.org/2026.findings-acl.523/.

Kutluhan Erol, James A. Hendler, and Dana S. Nau. HTN Planning: Complexity and Expressivity. In *AAAI Conference on Artificial Intelligence*, pp. 1123–1128, 1994. URL https://mlanthology.org/aaai/1994/erol1994aaai-htn/.

Tao Ge, Xin Chan, Xiaoyang Wang, Dian Yu, Haitao Mi, and Dong Yu. Scaling Synthetic Data Creation with 1,000,000,000 Personas, 2025. URL https://arxiv.org/abs/2406.20094.

Tao Ge, Baolin Peng, Hao Cheng, and Jianfeng Gao. Synthetic Computers at Scale for Long-Horizon Productivity Simulation, 2026. URL https://arxiv.org/abs/2604.28181.

Ilche Georgievski and Marco Aiello. HTN planning: Overview, comparison, and beyond. *Artificial Intelligence*, 222:124–156, May 2015. ISSN 0004-3702. doi: 10.1016/j.artint.2015.02.002. URL http://dx.doi.org/10.1016/j.artint.2015.02.002.

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 *COLM 2025*, 2025.

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 B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/edac78c3e300629acfe6cbe9ca88fb84-Paper-Conference.pdf.

Haoran Li, Qingxiu Dong, Zhengyang Tang, Chaojun Wang, Xingxing Zhang, Haoyang Huang, Shaohan Huang, Xiaolong Huang, Zeqiang Huang, Dongdong Zhang, Yuxian Gu, Xin Cheng, Xun Wang, Si Qing Chen, Li Dong, Wei Lu, Zhifang Sui, Benyou Wang, Wai Lam, and Furu Wei. Synthetic Data (Almost) from Scratch: Generalized Instruction Tuning for Language Models. *Transactions on Machine Learning Research*, 2025, July 2025. ISSN 2835-8856.

Yiyang Li, Zheyuan Zhang, Tianyi Ma, Zehong Wang, Keerthiram Murugesan, Chuxu Zhang, and Yanfang Ye. LongDA: Benchmarking LLM Agents for Long-Document Data Analysis, 2026. URL https://arxiv.org/abs/2601.02598.

Zeyao Ma, Bohan Zhang, Jing Zhang, Jifan Yu, Xiaokang Zhang, Xiaohan Zhang, Sijia Luo, Xi Wang, and Jie Tang. SpreadsheetBench: Towards Challenging Real World Spreadsheet Manipulation. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 94871–94908. Curran Associates, Inc., 2024. doi: 10.52202/079017-3007. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ac840df270ac537dd74530a15c332684-Paper-Datasets_and_Benchmarks_Track.pdf.

Mantas Mazeika, Alice Gatti, Cristina Menghini, Udari Madhushani Sehwag, Shivam Singhal, Yury Orlovskiy, Steven Basart, Manasi Sharma, Denis Peskoff, Elaine Lau, Jaehyuk Lim, Lachlan Carroll, Alice Blair, Vinaya Sivakumar, Sumana Basu, Brad Kenstler, Yuntao Ma, Julian Michael, Xiaoke Li, Oliver Ingebretsen, Aditya Mehta, Jean Mottola, John Teichmann, Kevin Yu, Zaina Shaik, Adam Khoja, Richard Ren, Jason Hausenloy, Long Phan, Ye Htet, Ankit Aich, Tahseen Rabbani, Vivswan Shah, Andriy Novykov, Felix Binder, Kirill Chugunov, Luis Ramirez, Matias Geralnik, Hernán Mesura, Dean Lee, Ed-Yeremai Hernandez Cardona, Annette Diamond, Summer Yue, Alexandr Wang, Bing Liu, Ernesto Hernandez, and Dan Hendrycks. Remote Labor Index: Measuring AI Automation of Remote Work, 2025. URL https://arxiv.org/abs/2510.26787.

Sebastian Nagel. CC-News, 2016. URL https://commoncrawl.org/blog/news-dataset-available.

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 Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 47717–47737. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/pan25g.html.

Tejal Patwardhan, Rachel Dias, Elizabeth Proehl, Grace Kim, Michele Wang, Olivia Watkins, Simon Fishman, Marwan Aljubeh, Phoebe Thacker, Laurance Fauconnet, Natalie Kim, Samuel Miserendino, Gildas Chabot, David Li, Patrick Chao, Michael Sharman, Alexandra Barr, Amelia Glaese, and Jerry Tworek. GDPval: Evaluating AI Model Performance on Real-World Economically Valuable Tasks. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 24005–24040, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/290c2430f91912204f30bbcc990fff1d-Paper-Conference.pdf.

Zeyu Qin, Qingxiu Dong, Xingxing Zhang, Li Dong, Xiaolong Huang, Ziyi Yang, Mahmoud Khademi, Dongdong Zhang, Hany Hassan Awadalla, Yi R. Fung, Weizhu Chen, Minhao Cheng, and Furu Wei. Scaling Laws of Synthetic Data for Language Models. In *COLM 2025*, July 2025. URL https://www.microsoft.com/en-us/research/publication/scaling-laws-of-synthetic-data-for-language-models/.

Shihao Rao, Liang Li, Jiapeng Liu, Guan Weixin, Xiyan Gao, Bing Lim, and Can Ma. Free your mouse! Command Large Language Models to Generate Code to Format Word Documents. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 16127–16142, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.902. URL https://aclanthology.org/2024.emnlp-main.902/.

Yiyou Sun, Xinyang Han, Weichen Zhang, Yuanbo Pang, Tianyu Wang, Yuhan Cao, Yixiao Huang, Chris Duroiu, Haoyun Zhang, Jeffrey Lin, Weishu Zhang, Tyler Zeng, Ying Yan, Bo Liu, Hanson Wen, Mingyang Xu, Xiaoyuan Liu, Zimeng Chen, Weiyan Shi, Amanda Dsouza, Vincent Sunn Chen, Patrick Bryant, Carl Boettiger, Yamini Rangan, Bradley Rothenberg, Kyle Steinfeld, Arvind Rao, Tapio Schneider, Georgios Yannakakis, Laure Zanna, Kaan Ozbay, Ida Sim, Tarek Zohdi, George Em Karniadakis, Jack Gallant, Teresa Head-Gordon, Yushan Li, Wenxi Deng, Tao Sun, Huiqi Wang, Zhun Wang, Justin Xu, Chris Yuhao Liu, Yafei Cheng, Rongwang Hu, Aras Bacho, Shengcao Cao, Zengyi Qin, Yixiong Chen, Hengduan Fan, Hao Liu, Lin Zeng, Shashank Muralidhar Bharadwaj, Litian Gong, Yingxuan Yang, Maojia Song, Ruheng Wang, Zongzheng Zhang, Honglin Bao, Shuo Lu, Jianhong Tu, Zhonghua Wang, Zheng Zhang, Zijiao Chen, Yanqiong Jiang, Zhendong Li, Bohan Lyu, Chang Ma, Peiran Xu, Benran Zhang, Shangding Gu, Haoyue Hua, Haoyang Li, Wanzhe Liao, Chengzhi Liu, Junbo Peng, Haoran Sun, Zechen Xu, Bo Chen, Jiayi Cheng, Yi Jiang, Keying Kuang, Yuan Li, Youbang Pan, Ziyan Rao, Alexander Schubert, Yifan Shen, Vincent Siu, Xiatao Sun, Kangqi Zhang, Xiaopan Zhang, Yuchen Zhu, Ishaan Singh Chandok, Lei Ding, Jingxuan Fan, Andrew Glover, Jiaming Hu, Yiran Hu, Wenbo Huang, Zixin Jiang, Haoran Jin, Lukas Kim, Ming Liu, Yang Liu, Alireza Rafiei, Xuhuan Shen, Kunyang Sun, Sophia Sun, Ting Sun, Eric Wang, Yixin Wang, Hanwen Xing, Sihan Xu, Yuzheng Xu, Zhongxing Xu, Zhiling Yan, Boqin Yuan, Ruiqi Zhang, Yifan Zhang, Zibo

Zhao, Liana, Santanu Bosu Antu, Haoyue Bai, Carlo Bosio, Joseph Cavanagh, Patricia Cavazos-Rehg, Tianxing Chen, Xuewen Chen, Yipu Chen, Chenyu Zhu, Chen Dai, Stefano De Castro, Yunfu Deng, Kaustubh Dhole, Jiayuan Ding, Chenchen Du, Zhehang Du, Hao Fan, Run-Ze Fan, Hengyu Fu, Shi Gu, Yifan Gu, Charlie Guo, Baihe Huang, Baixiang Huang, Rimika Jaiswal, Zhihan Jiang, Ran Jin, Erin Kasson, Xin Lan, Joseph Lee, Deren Lei, Chenyu Li, Daofeng Li, Haitao Li, Hongwei Li, Jingyan Li, Xiao Li, Yi Li, Yinsheng Li, Yuangang Li, Zhixu Li, Wenyu Liang, Longtai Liao, Kevin Qinghong Lin, Andy Zeyi Liu, Che Liu, Jiaming Liu, Kaiyuan Liu, Xuan Liu, Pan Lu, Wenbo Lv, Yicheng Lyu, Qiuyang Mang, Kyle Montgomery, Yuzhou Nie, Ruoxi Ning, Jorin Overwiening, Xu Pan, Layna Paraboschi, Core Francisco Park, Justin Purnomo, Swati Rajwal, Scott Rankin, Bixuan Ren, Yiren Rong, HaoYang Shang, Ventus Shaw, Fiona Shen, Jiawei Shen, Minqi Shi, Shi Qiu, Huaxiu Yao, Tianneng Shi, Jonah So, Vladislav Susoy, Hannah Szlyk, Haocheng Wang, Jialu Wang, Wei Wang, Xinyu Wang, Zehao Wang, Dowling Wong, Angela Wu, Dehao Wu, Fangyu Wu, Mengyuan "Millie" Wu, Yu Wu, Yuchen Wu, Yuhao Wu, Qingpo Wuwu, Weihang Xiao, Yongyi Xiong, Fan Xu, Ruiling Xu, Mingxuan Yan, Benjamin Yang, Jirong Yang, Sen Yang, Xiaoli Yang, Yushi Yang, Haoran Ye, Xiaohu Yu, Zhengming Yu, Chenlong Zhang, Chi Zhang, Hanning Zhang, Hanwen Zhang, Junge Zhang, Kunpeng Zhang, Song Zhang, Wenjin Zhang, Wenshuo Zhang, Ying Zhang, Yizhi Zhang, Brian Zhao, Qijian Zhao, Yimin Zhao, Yuhaohua Zheng, Liwei Zhou, Tianyue Zhou, Sichen Zhu, Siqi Zhu, Yan Zhu, Yishu Zhu, Jierui Zuo, Chonghao Cai, Helena Casademunt, Wenjia Chen, Cheng Cheng, Nawen Deng, Rao Fu, Tianfu Fu, Yifan Han, He Ren, Zhenyu He, Qiao Jin, Langlang Li, Yuetai Li, Sylvia Liu, Lu Lu, Luqing Zhou, Subhabrata Mukherjee, Yunqi Ouyang, Yin Ren, Dawei Shi, Haoran Wu, Zhiyue Wu, Hannah Yao, Zhuoran Yi, Jenny Yu, Rhea Zhan, Hang Zhou, Blake Zhu, Junfan Zhu, Alan Yuille, Yang Liu, Russell Alan Poldrack, Jiachen Li, Zhenglu Li, Molei Tao, Jing Huang, Wenqi Shi, Costas Spanos, Lichao Sun, Chenguang Wang, Orson Xu, Zhen Dong, Hector Gomez, Aylin Caliskan, Ali Emami, Haimin Hu, Zhi Li, Lihui Liu, Murphy Niu, Yi Shao, Jianxin Sun, Mikko Tolonen, Ting Wang, Sanjiv Das, Yanjun Gao, Wenbo Guo, Erika J Schneider, Zhiyong Lu, Yian Ma, Mark Mueller, Radha Poovendran, Somayeh Sojoudi, Yinglun Zhu, and Dawn Song. Agents' Last Exam, 2026. URL https://arxiv.org/abs/2606.05405.

Harsh Trivedi, Tushar Khot, Mareike Hartmann, Ruskin Manku, Vinty Dong, Edward Li, Shashank Gupta, Ashish Sabharwal, and Niranjan Balasubramanian. AppWorld: A Controllable World of Apps and People for Benchmarking Interactive Coding Agents. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16022–16076, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.850. URL https://aclanthology.org/2024.acl-long.850/.

Dunwei Tu, Hongyan Hao, Hansi Yang, Yihao Chen, Yi-Kai Zhang, Zhikang Xia, Yu Yang, Yueqing Sun, Xingchen Liu, Furao Shen, Qi Gu, Hui Su, and Xunliang Cai. ScaleEnv: Scaling Environment Synthesis from Scratch for Generalist Interactive Tool-Use Agent Training, 2026. URL https://arxiv.org/abs/2602.06820.

Bertie Vidgen, Austin Mann, Abby Fennelly, John Wright Stanly, Lucas Rothman, Marco Burstein, Julien Benchek, David Ostrofsky, Anirudh Ravichandran, Debnil Sur, Neel Venugopal, Alannah Hsia, Isaac Robinson, Calix Huang, Olivia Varones, Daniyal Khan, Michael Haines, Austin Bridges, Jesse Boyle, Koby Twist, Zach Richards, Chirag Mahapatra, Brendan Foody, and Osvald Nitski. APEX-Agents, 2026. URL https://arxiv.org/abs/2601.14242.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754/.

Zhaoyang Wang, Canwen Xu, Boyi Liu, Yite Wang, Siwei Han, Zhewei Yao, Huaxiu Yao, and Yuxiong He. Agent World Model: Infinity Synthetic Environments for Agentic Reinforcement Learning. In *ICML 2026*, February 2026. URL https://www.microsoft.com/en-us/research/publication/agent-world-model-infinity-synthetic-environments-for-agentic-reinforcement-learning/.

Yuxiang Wei, Zhe Wang, Jiawei Liu, Yifeng Ding, and Lingming Zhang. Magicoder: Empowering Code Generation with OSS-Instruct. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 52632–52657. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/wei24h.html.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering Large Pre-Trained Language Models to Follow Complex Instructions. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 30745–30766, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/82eec786fdfbbfa53450c5feb7d1ac92-Paper-Conference.pdf.

Frank (Fangzheng) Xu, Yufan Song, Boxuan Li, Yuxuan Tang, Kritanjali Jain, Mengxue Bao, Zora Wang, Xuhui Zhou, Zhitong Guo, Murong Cao, Mingyang Yang, Hao Yang Lu, Amaad Martin, Zhe Su, Leander Maben, Raj Mehta, Wayne Chi, Lawrence Jang, Yiqing Xie, Shuyan Zhou, and Graham Neubig. TheAgentCompany: Benchmarking LLM Agents on Consequential Real World Tasks. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-0315. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0d744742f6fac4d1134c019b7cef3c8a-Paper-Datasets_and_Benchmarks_Track.pdf.

John Yang, Kilian Lieret, Carlos Jimenez, Alexander Wettig, Kabir Khandpur, Yanzhe Zhang, Binyuan Hui, Ofir Press, Ludwig Schmidt, and Diyi Yang. SWE-smith: Scaling Data for Software Engineering Agents. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference. Curran Associates, Inc., 2025. doi: 10.52202/085713-3239. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/8b86cf5ace600c48fd188efbb8dedec8-Paper-Datasets_and_Benchmarks_Track.pdf.

Darvin Yi, Teng Liu, Mattie Terzolo, Lance Hasson, Ayan Sinha, Pablo Mendes, and Andrew Rabinovich. UpBench: A Dynamically Evolving Real-World Labor-Market Agentic Benchmark Framework Built for Human-Centric AI, 2025. URL https://arxiv.org/abs/2511.12306.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 Embedding: Advancing Text Embedding and Reranking Through Foundation Models, 2025. URL https://arxiv.org/abs/2506.05176.

Ziyun Zhang, Zezhou Wang, Xiaoyi Zhang, Zongyu Guo, Jiahao Li, Bin Li, and Yan Lu. InfiniteWeb: Scalable Web Environment Synthesis for GUI Agent Training. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 28465–28492, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1313. URL https://aclanthology.org/2026.acl-long.1313/.

A TASK SYNTHESIS

We give the core prompt instructions and structural checks used to construct task hierarchies and their input materials.

A.1 OCCUPATION RELEVANCE

Occupational relevance depends on what an article discusses, not whether it names the occupation. We apply the criterion below both when training classifiers and when selecting articles for task generation. In the retrieval study, we label 2,000 articles per occupation in each of ten rounds. Round 0 uses the top retrieved articles from the encoded subset, while rounds 1 to 9 sample the predicted positive pools from the full corpus. Refinement stops when the estimated precision reaches 99% or after round 9.

Retrieval query

What is happening in the professional world of {occupation} in the sector of {sector_title}: the work they do, the field and surrounding context they engage with, and anything that touches or relates to their practice, wherever this occupation operates?

Relevance criterion

For each article, judge one thing: does it connect to this occupation's professional world, the work they do and the field and context around it?

An article belongs (true) when its substance touches that world in any real way: the work and how it is done, the methods, tools, and technology, the cases and situations, the trends shaping the field, the people and organizations in it, or the wider context and surroundings a practitioner works within. It need not be written for an expert, and it need not be about the work itself; it is enough that the field or the world this occupation genuinely operates in is part of what the article is about.

An article does not belong (false) only when the occupation is named in passing with no real connection to its world, or when the substance is generic enough to belong to any field or occupation just as well. If an article genuinely relates to this occupation's world, prefer true.

A.2 TASK HIERARCHY GENERATION

Parent-child links describe task decomposition, while sibling dependencies specify execution order. We define these relations and the human-work time horizon in the prompt as follows.

Task relations

Keep two relations distinct. Composition is vertical: a parent is made of its children. Order is horizontal and only among siblings: within one parent's children, some must finish before others begin, while the rest are independent. Record order in `depends_on` (sibling ids that must complete first), and omit it when work is parallel. Order never crosses branches; if two pieces truly constrain each other, make them siblings under a common parent.

Time horizon

`horizon`: this piece's natural time scale, one of "month", "week", "day", "hour". A parent is at least as coarse as its children, but scales may skip: a "week" node may hold a genuinely small "hour" child. Let it reflect real size.

Table 3: Inputs required for GDPval grading. All methods receive the task, input materials, and candidate submission.

Method	Reference submission	Human-written rubric	Grading output
Pairwise	Required	Not required	Relative preference
Human rubric	Not required	Required	Normalized criterion score
Market value	Not required	Not required	Normalized deliverable value

A.3 Shared Input Material Construction

We construct input materials for the whole task hierarchy, allowing several tasks to share the same files. The instructions below specify how these files are shared and obtained through web retrieval or synthesis from searched information.

Material sharing

Each file holds more than any single task needs, so that several tasks can share it, each drawing the slice it uses. Never make one file per task, and never tailor a file to a task's answer.

Retrieval and synthesis

Search the open web (`web_search`) for authoritative sources such as agencies, standards bodies, companies, datasets, and guidelines, and take the real thing when you can get it. When you cannot obtain a full source, gather its real substance (figures, terms, structures, thresholds, names) and author a complete file grounded in those facts; the result is believable and rooted in reality, never a thin sketch and never a tailored key.

A.4 Structural Checks

We check that each material reference points to an existing file or a nonempty folder and has a source record. The construction agent receives detected material errors for correction before the materials are accepted. Before execution, we also validate task identifiers and dependency references and reject cycles in the dependency graph. These checks catch missing inputs and invalid execution orders, but do not verify the correctness of file contents.

B Training and Evaluation Details

B.1 Training Data

We retain a teacher run only if it finishes and all specified deliverables are present. Expected file paths must match collected files, and expected folders must contain at least one file. This excludes runs that end without producing the required outputs, but does not judge their quality.

During long executions, context compaction summarizes the preceding conversation so the teacher can continue from the task instructions and the summary. We export conversation windows as interaction samples, including tool interactions and available images, and form separate samples for generating the summaries. The data thus supervise both task execution and summarization of earlier work. The 100K tasks yield 191,327 interaction windows and 32,754 summary samples, totaling 224,081 exported samples. We report dataset size in distinct tasks because one task can produce multiple samples. For the scaling experiments, we randomly sample task subsets from the 100K dataset and keep the fine-tuning settings fixed.

B.2 Evaluation Scores

Table 3 compares the additional inputs needed for GDPval grading. Pairwise grading requires a reference submission, whereas human-rubric grading requires human-written criteria. Market value grading requires neither, scoring submissions relative to the estimated task price.

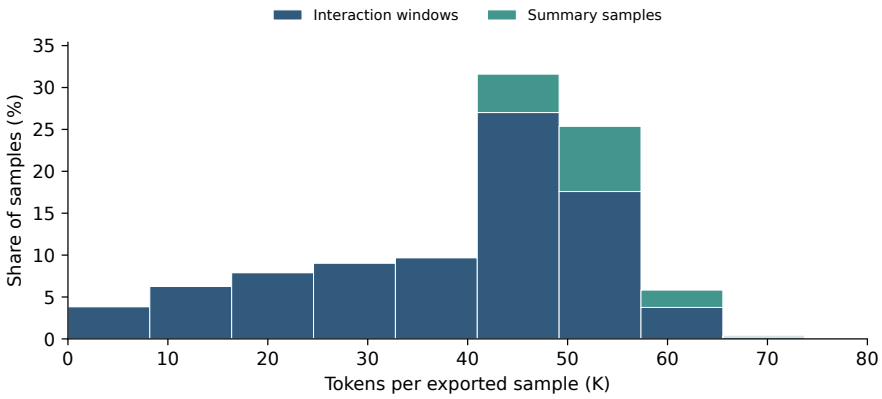


Figure 6: Token lengths of 224,081 exported samples, shown up to 80K tokens. Bars separate interaction windows and summary samples.

Human-rubric scores normalize the weighted criterion total over its possible range, including penalties. For market value, GPT-5.5 estimates the price of completing the requested work, excluding the value of materials already supplied. It receives the task instructions, occupation, sector, and input file list, and we average three estimates to set a fixed task price in USD. A GPT-5.5 grading agent then inspects the submission and estimates its value. We divide this value by the task price and cap the score at 100%. These are model estimates, not observed market prices.

We average normalized scores across tasks within each trial and then across trials. For market value, each task thus has equal weight regardless of its estimated price. We compute win rates separately by comparing model and human scores on jointly scored tasks. Human submissions are used for this comparison, not for grading individual deliverables.

For APEX-Agents, we average criterion scores within each task and then across tasks, so tasks with more criteria do not receive more weight. Mean score captures partial success, whereas Pass@1 counts only tasks that satisfy all graded criteria.

C Dataset Analysis

C.1 Training Sample Lengths

We measure exported samples with the training tokenizer and conversation template, counting tool descriptions, tool interactions, and image tokens. Figure 6 shows a median length of 46,387 tokens and a 95th percentile of 58,131 tokens. The typical sample therefore spans tens of thousands of tokens, while the 95th percentile remains below half the configured SFT maximum of 131,072 tokens. Only two samples exceed that maximum, with the longest reaching 145,222 tokens. These lengths describe individual exported windows or summary samples, not full task executions or the final sequences processed during training.

C.2 Task Horizons and Execution Lengths

The 100K retained tasks include leaf nodes (78.9%) and parent nodes (21.1%), retaining both component tasks and broader assignments. As shown in Figure 7, hour and day tasks account for 80.0% of the data. Longer horizons are associated with more interaction, not just more generated text. From hour to month tasks, median teacher output increases from 14.1K to 50.0K tokens, while median interaction turns increase from 11 to 31. The horizon labels describe expected human effort, not measured completion times. The four reported categories cover 99,976 tasks, of which 99,966 have available execution records.

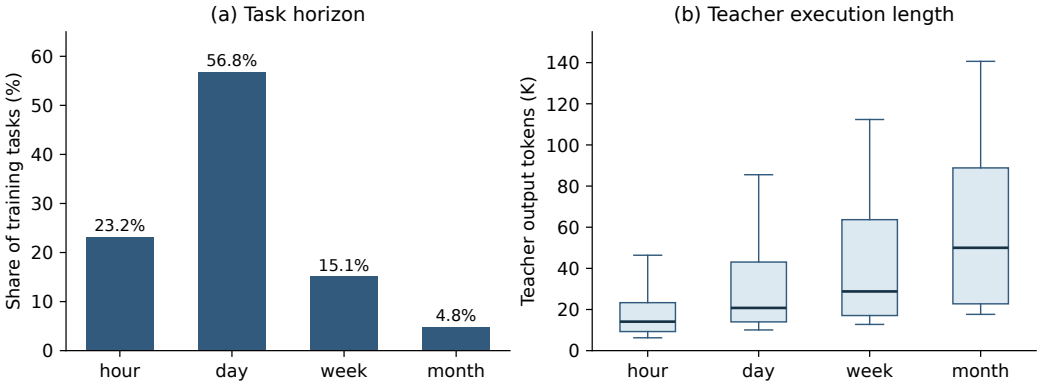


Figure 7: Task horizons and teacher execution lengths. (a) Task shares use all 100K tasks as the denominator. (b) Output tokens cover each task's full execution. Boxes show quartiles and medians, and whiskers show the 10th and 90th percentiles.

Table 4: Statistics for the 100 task hierarchies in the construction-cost study. An input material item is a referenced file or folder; a task reference is one assignment of an item to a task.

Statistic	Mean	Median
Tasks per hierarchy	25.40	25
Distinct input material items per hierarchy	7.55	7
Task references to input materials per hierarchy	59.44	53

C.3 Material Sharing

Table 4 summarizes the 100 task hierarchies used in the construction-cost study. Each hierarchy contains 25.4 tasks on average, sharing 7.55 distinct input material items through 59.44 task references. The cost advantage comes from reusing constructed materials across these assignments, rather than providing a separate set for each task.

C.4 Benchmark Overlap

After training, we compare the 100K task instructions with all 220 tasks in the public GDPval gold subset and the 58 APEX-Agents file-deliverable tasks. After text normalization, we find no exact matches or shared sequences of 13 consecutive words between training and benchmark instructions. We also find no exact duplicate instructions within the training set. These results provide evidence against verbatim copying of benchmark instructions, but do not rule out paraphrases or overlap in input materials and deliverables.

D Additional Results

D.1 Occupation-Level Retrieval Results

Between rounds 1 and 9, retrieval precision improves across all 44 occupations, as shown in Figure 8. Both rounds evaluate random samples from full-corpus predicted positive pools, whereas round 0 evaluates the top retrieved articles from the encoded subset. Retail has the lowest sector mean in round 1 and improves from 67.8% to 89.4% mean precision, compared with 90.6% to 93.8% in health care. Across the nine sectors, the gap between the largest and smallest sector means falls from 22.9 to 9.0 percentage points. Precision and candidate-pool size both increase in 41 of the 44 occupations, and every occupation retains more than one million candidates in the final round.

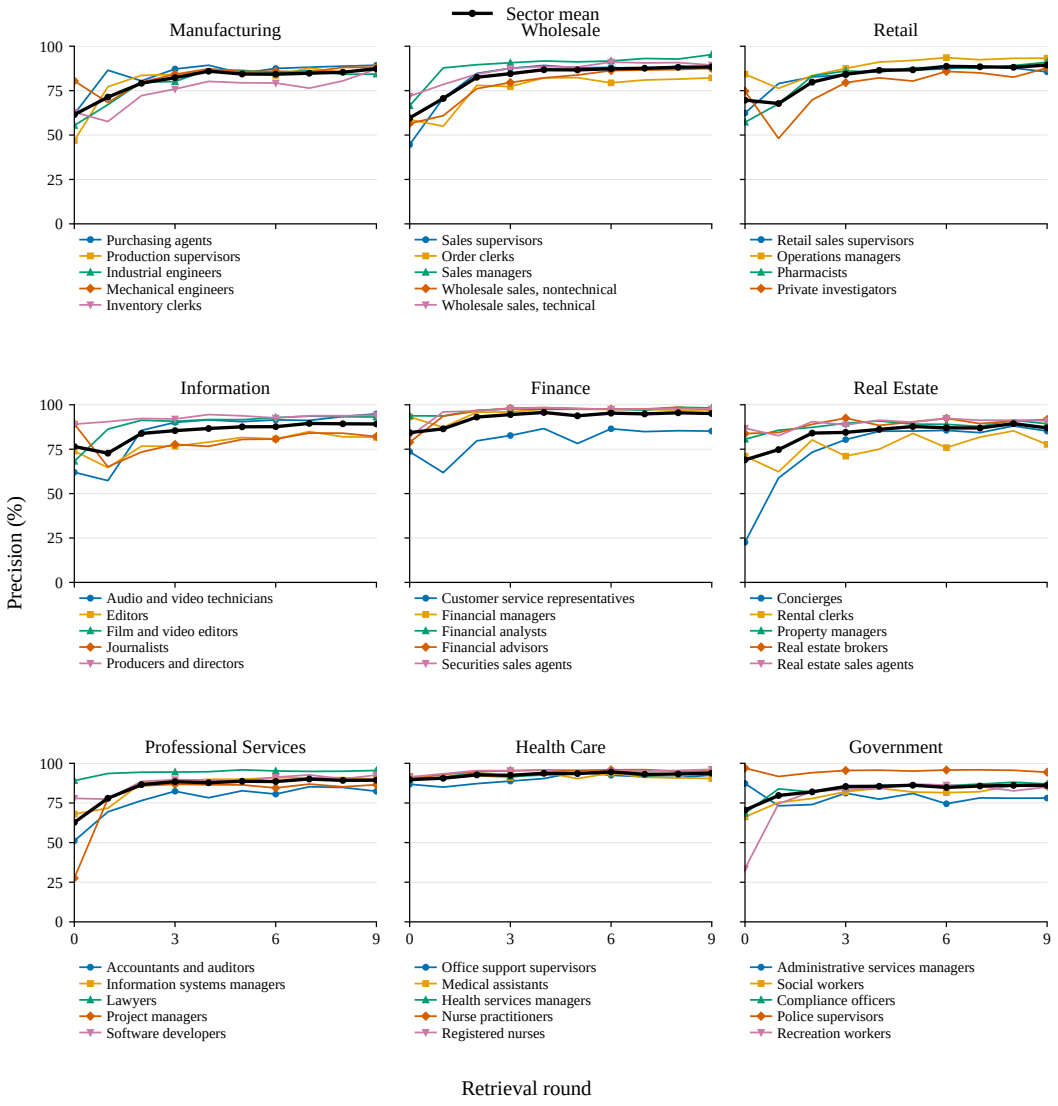


Figure 8: Occupation-level retrieval precision across nine sectors. Each panel shows the occupations named below it and their sector mean. Round 0 is zero-shot retrieval, and rounds 1 through 9 evaluate full-corpus predicted positive pools.

Table 5: Performance of Qwen3.5-4B-Base fine-tuned on trajectories from 10K tasks solved by each teacher. Parentheses show percentage-point gains over the base model.

	GDPval (%)		APEX-Agents (%)	
Teacher model	Win	Win or tie	Pass@1	Mean score
Qwen3.6-27B	28.0 (+21.4)	35.2 (+25.3)	5.2 (+1.8)	26.5 (+13.1)
GLM-5.2	33.9 (+27.3)	39.5 (+29.6)	5.2 (+1.8)	27.1 (+13.7)

D.2 Teacher Comparison

Table 5 compares the same 4B base model fine-tuned on trajectories from 10K tasks solved by each teacher. Both teachers improve all four metrics and approximately double the base model's APEX mean score. The gains with GLM-5.2 show that training does not require a teacher from the

Table 6: Scores and win rates on GDPval. Win rates compare model and human submissions.

	Pairwise	Human rubric		Market value	
Model	Win (%)	Score (%)	Win (%)	Score (%)	Win (%)
Qwen3.6-27B	77.5	81.9	50.9	61.4	72.6
Qwen3.5-9B	26.7	63.5	21.5	28.2	19.8
Qwen3.5-4B	20.9	61.0	15.4	21.9	11.0
Qwen3.5-9B-Base	14.3	53.0	9.9	18.8	9.2
Qwen3.5-4B-Base	6.6	48.1	7.3	13.0	4.5

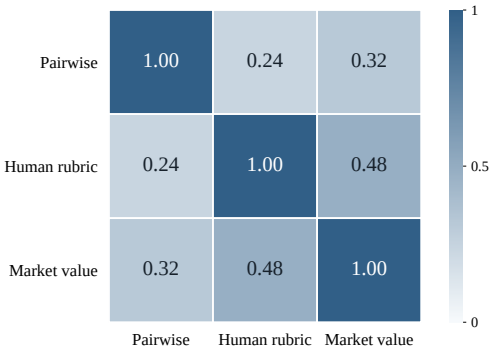


Figure 9: Task-level Spearman correlations between evaluation methods, averaged across the five models. Diagonal entries are self-correlations.

Qwen family. This comparison does not isolate teacher quality because the GLM-5.2 teacher has no image-viewing tool during trajectory generation.

D.3 Agreement Across Evaluation Methods

All three methods rank the five models in the same order, as shown in Table 6. To compare task-level scores, we compute Spearman correlations across jointly scored tasks within each model and average the coefficients across models. For this analysis, pairwise outcomes are encoded as 1 for a win, 0.5 for a tie, and 0 for a loss; the other methods retain their normalized scores.

As shown in Figure 9, market value correlates more closely with human-rubric scores than pairwise outcomes do, with mean correlations of 0.48 and 0.24, respectively. This difference holds for each of the five models. Market value and pairwise outcomes have a mean correlation of 0.32. The closer agreement with human-rubric scores supports studying market value as training feedback that does not require reference submissions or human-written rubrics. Its training value remains untested.