

Learning Efficient Tool Orchestration with Language Models

Stanford CS224N {Custom}

Orhun Akengin

Department of Computer Science
Stanford University
akengin@stanford.edu

Abstract

In this project, we study how to train language models to efficiently orchestrate tools and models for multi-step problem solving. We fine-tune Qwen3-8B and Qwen3-32B base models on 50k trajectories. Our experiments show that SFT orchestrator improves accuracy by 10.7 points on DeepResearch and 3.5 points on SimpleQA compared to an untrained orchestrator baseline. Our RL-based orchestrator further improves results, reaching up to a 19.6 point gain on DeepResearch and a 14.0 point gain on SimpleQA with the 32B model.

1 Key Information to include

- TA mentor: David Anugraha
- External collaborators: Herumb Shandilya (MS collaborator)
- External mentor: Azalia Mirsoheini (faculty), Jon Saad Falcon (supervisor)
- Sharing project: Yes, for CS 234

2 Introduction

Language models are increasingly used as agents that solve tasks through multiple steps and by interacting with external tools. Instead of relying on a single large model for every step, an orchestrator model can decide which tool or model to use at each stage of the process. This can make systems more efficient while still maintaining strong task performance. In this project, we study how to train language models to act as tool orchestrators for multi-step problem solving.

3 Related Work

Our work mostly builds up on ToolOrchestra [1]. Unlike their RL-only setup, we are exploring different orchestrators trained using SFT with LoRA and an RL framework with custom designed multi-objective reward that additionally optimizes energy and peak power compared to their work. Instead of using their datasets, we generated our own large-scale tool usage trajectories with telemetry and train separate SFT and RL orchestrators to compare which performs better.

4 Approach

4.1 Main Approach

Our framework consists of three parts: trajectory generation, orchestrator training, and evaluation. For trajectory generation, we used an already existing data generation pipeline to create a vast dataset, and then filtered it to be used for training. 1

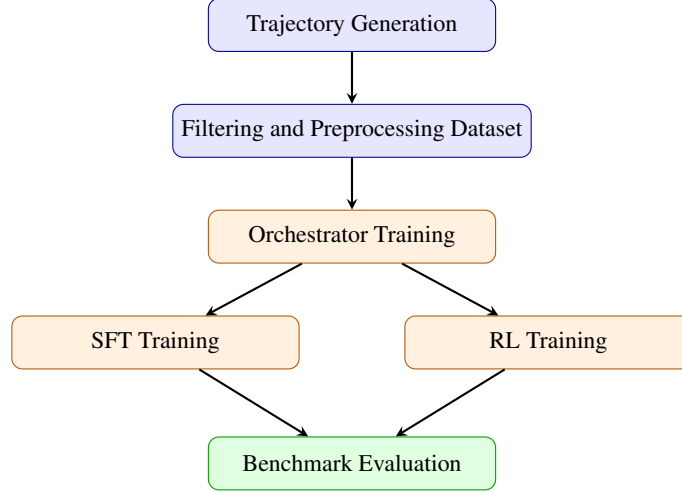


Figure 1: Our pipeline: trajectory generation, orchestrator training, and benchmark evaluation.

In the scope of this project, we study two paradigms for orchestrator training. (1) Supervised fine-tuning (SFT) with LoRA adapters on frozen base model weights, (2) A GRPO-based reinforcement learning setup that optimizes a multi-objective reward. For both of the setups, we used Qwen3-8B and Qwen3-32B base models to fine tune. For the GRPO based RL setup, we used a custom reward function shown in Figure 2

$$R = \alpha \cdot \text{accuracy} - \beta_c \cdot \text{cost} - \beta_l \cdot \text{latency} - \beta_e \cdot \text{energy} - \beta_p \cdot \text{power}$$

Figure 2: Multi-objective reward used for RL training.

The reward combines task accuracy with penalties for resource usage to encourage efficient orchestration decisions. Accuracy incentivizes correct task completion, while the cost, latency, energy, and power terms discourage unnecessary tool calls and promote selecting cheaper or faster tools when possible.

While training, we ran WandB sweeps on hyperparameters like LR, LoRA rank and epochs to search the hyperparameter space and find the optimal values. Our pipeline selects the best checkpoint by validation performance and runs full evaluation on all benchmarks we have.

For evaluation, we used 4 public benchmarks 3 that includes various questions from different domains. Because of the time constraints, we only used a small portion of each benchmark for evaluations.

4.2 Baselines

We compare our results against the following baselines:

- **Untrained Orchestrator:** The base model (Qwen3-8B or Qwen3-32B [2]) prompted in orchestration format without any fine-tuning.
- **Instruct Version of Base Model:** The same base model but in instruct version used to solve tasks directly without any tools.

We then evaluate all methods on a portion of four well-known agentic benchmarks: SimpleQA [3], GAIA [4], HLE [5], and DeepResearch [6].

4.3 Code Base

The orchestration pipeline including trajectory generation, telemetry collection, SFT training, GRPO framework, and evaluation was developed in our lab environment. LoRA training is implemented using HuggingFace Transformers and PEFT. I was responsible for implementing the LoRA training pipeline and the full benchmark evaluation framework. We relied on the generative AI tools to create the bone structure of our code base.

5 Experiments

5.1 Data

We train the orchestrator using trajectories generated by our internal end-to-end pipeline. Given a task and tool set, a teacher model executes multi-step tool interactions while logging actions, tool outputs, and efficiency telemetry (cost, energy, latency, peak power). Thus far, we generated 168k trajectories. We then filter them using criteria such as requiring at least 2 turns and the use of multiple tools, resulting in 53k high-quality episodes. For the training, we used these 53k trajectories with a 10% validation split.

5.2 Evaluation method

We mainly evaluate on four public agentic benchmarks 3: SimpleQA [3], GAIA [4], HLE [5], and DeepResearch [6]. SimpleQA consists of short factual questions requiring precise answers. GAIA and HLE require multi-step reasoning and tool use across diverse domains. DeepResearch involves long-form research-style queries evaluated against reference articles. For SimpleQA, GAIA, and HLE we report exact-match accuracy. For DeepResearch, we use an LM-based judge that scores responses according to a rubric against reference materials. We used 200 tasks from each benchmark instead of using all of the tasks due to time and computational constraints.

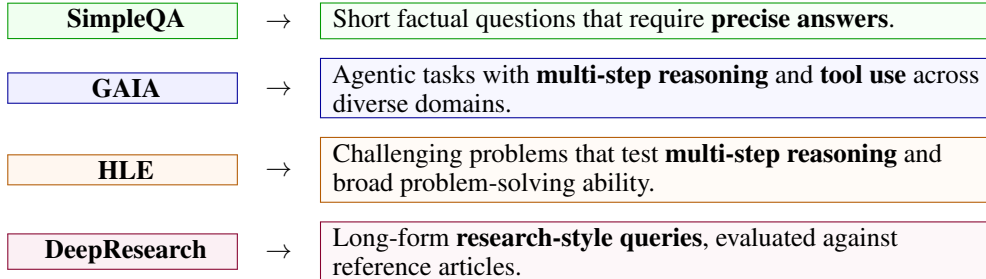


Figure 3: Benchmarks used to evaluate orchestration performance.

5.3 Experimental details

We train orchestrator models using both supervised fine-tuning (SFT) and reinforcement learning (RL) on Qwen3-8B and Qwen3-32B base models.

For SFT, we fine-tune the models using LoRA adapters while freezing base weights and training with AdamW. Hyperparameters such as learning rate, LoRA rank, batch size, and context length were selected using WandB sweeps. The final configuration is summarized in Figure 4.

For RL, we train the orchestrator using a GRPO-based policy optimization setup with AdamW and a multi-objective reward. We used a small learning rate, grouped rollouts per prompt, and KL regularization to stabilize training. The detailed RL configuration is shown in Figure 4.

SFT Training Setup

```
method: SFT with LoRA
optimizer: AdamW
learning_rate: 4e-5
lora_rank: 128
lora_alpha: 256
batch_size: 8
context_length: 16384
epochs: 1
train/val_split: 90/10
training_time: 6-8 hrs / model
```

RL Training Setup

```
method: GRPO
optimizer: AdamW
learning_rate: 1e-6
batch_size: 16
group_size: 8
max_prompt_length: 24000
max_response_length: 8768
kl_coef: 1e-4
num_epochs: 10
clip_ratio: 0.2
max_grad_norm: 1.0
training_time: 12-16 hrs /
model
```

Figure 4: Training configurations for SFT and RL orchestrators.

6 Results

Figure 5 summarizes the benchmark results for the baseline systems and our trained orchestrators. Overall, trained orchestrators outperform the baselines on most tasks. The largest improvements appear on DeepResearch and SimpleQA for both model sizes. Gains on GAIA are relatively small, while performance on HLE decreases compared to the untrained orchestrator. RL-based training provides additional improvements over SFT on several benchmarks.

6.1 Analysis

If we compare SFT and RL orchestrators more closely, we can see that RL consistently produces small improvements over SFT across most benchmarks. For the 8B model, RL increases DeepResearch from 0.37 to 0.39 and SimpleQA from 0.24 to 0.26, while slightly improving GAIA and partially recovering the drop seen on HLE. A similar pattern appears for the 32B model, where RL improves DeepResearch from 0.42 to 0.44 and SimpleQA from 0.29 to 0.31.

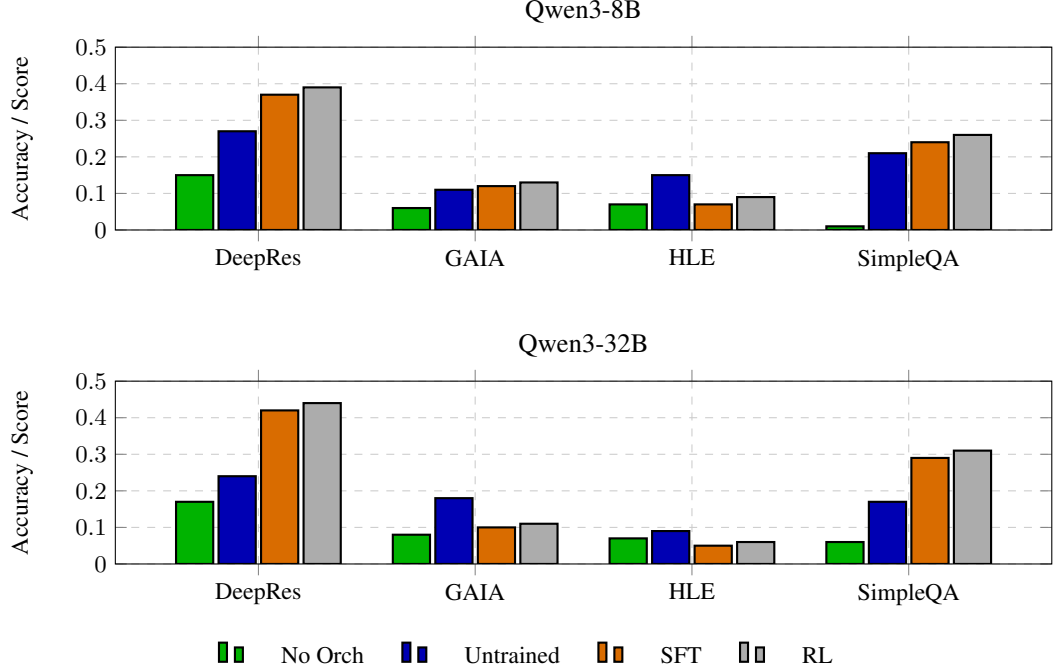
These differences arise from the training objectives of the two methods. SFT learns to imitate trajectories from the dataset, while RL directly optimizes task success through the reward function. This allows RL to refine orchestration decisions such as tool selection and stopping behavior. However, the improvements remain relatively small, which suggests that the SFT model already captures much of the useful behavior present in the trajectories and that RL mainly provides incremental refinement rather than large behavioral changes.

7 Conclusion & Future Work

In this project, we trained LMs to act as tool orchestrators for multi-step problem solving using both SFT and RL. Our results show that trained orchestrators can improve performance on several benchmarks, with RL providing a better gains over SFT.

Limitations Due to time and resource constraints, we evaluated only 200 tasks from each benchmark rather than the full datasets. Our experiments also used limited hyperparameter tuning and only two base models (Qwen3-8B and Qwen3-32B). More extensive tuning and testing with additional models could further improve and better validate the results.

Future work We plan to explore improved reward functions and further study the differences between SFT and RL-based orchestrators. We also plan to scale the system to larger tool ecosystems and run evaluations on full benchmark datasets.



8B Model					32B Model				
Benchmark	No Orch	Untrained	SFT	RL	Benchmark	No Orch	Untrained	SFT	RL
DeepResearch	0.15	0.27	0.37	0.39	DeepResearch	0.17	0.24	0.42	0.44
GAIA	0.06	0.11	0.12	0.13	GAIA	0.08	0.18	0.10	0.11
HLE	0.07	0.15	0.07	0.09	HLE	0.07	0.09	0.05	0.06
SimpleQA	0.01	0.21	0.24	0.26	SimpleQA	0.06	0.17	0.29	0.31

Figure 5: Benchmark accuracy for baseline systems, SFT orchestrators, and RL orchestrators.

References

- [1] NVIDIA Research. Toolorchestra: A framework for benchmarking and training multi-tool large language model agents. <https://research.nvidia.com/labs/lpr/ToolOrchestra/>, 2024. Accessed: 2026-02.
- [2] Qwen Team. Qwen3 technical report, 2025.
- [3] OpenAI. Simpleqa: A benchmark for measuring short-form factuality in large language models. <https://cdn.openai.com/papers/simpleqa.pdf>, 2024.
- [4] Grégoire Mialon, Clémentine Fourier, Thomas Wolf, Yann LeCun, and Thomas Scialom. Gaia: A benchmark for general ai assistants. In *International Conference on Learning Representations*, 2023.
- [5] Jinghui Zhou, Ethan Perez, Percy Liang, et al. Humanity’s last exam: A benchmark for evaluating advanced reasoning in language models. *arXiv preprint*, abs/2501.14249, 2025.
- [6] Mingxuan Du, Benfeng Xu, Chiwei Zhu, Xiaorui Wang, and Zhendong Mao. Deepresearch bench: A comprehensive benchmark for deep research agents. *arXiv preprint*, 2025.