CSCI 5942: AI ENGINEERING, LECTURE 21

Prompt Engineering and Tools

Required

Khattab et al., DSPy · arXiv:2310.03714

Optional

Wang et al., Self-Consistency · arXiv:2203.11171

Christoffer Heckman
Department of Computer Science, University of Colorado Boulder

November 5, 2026

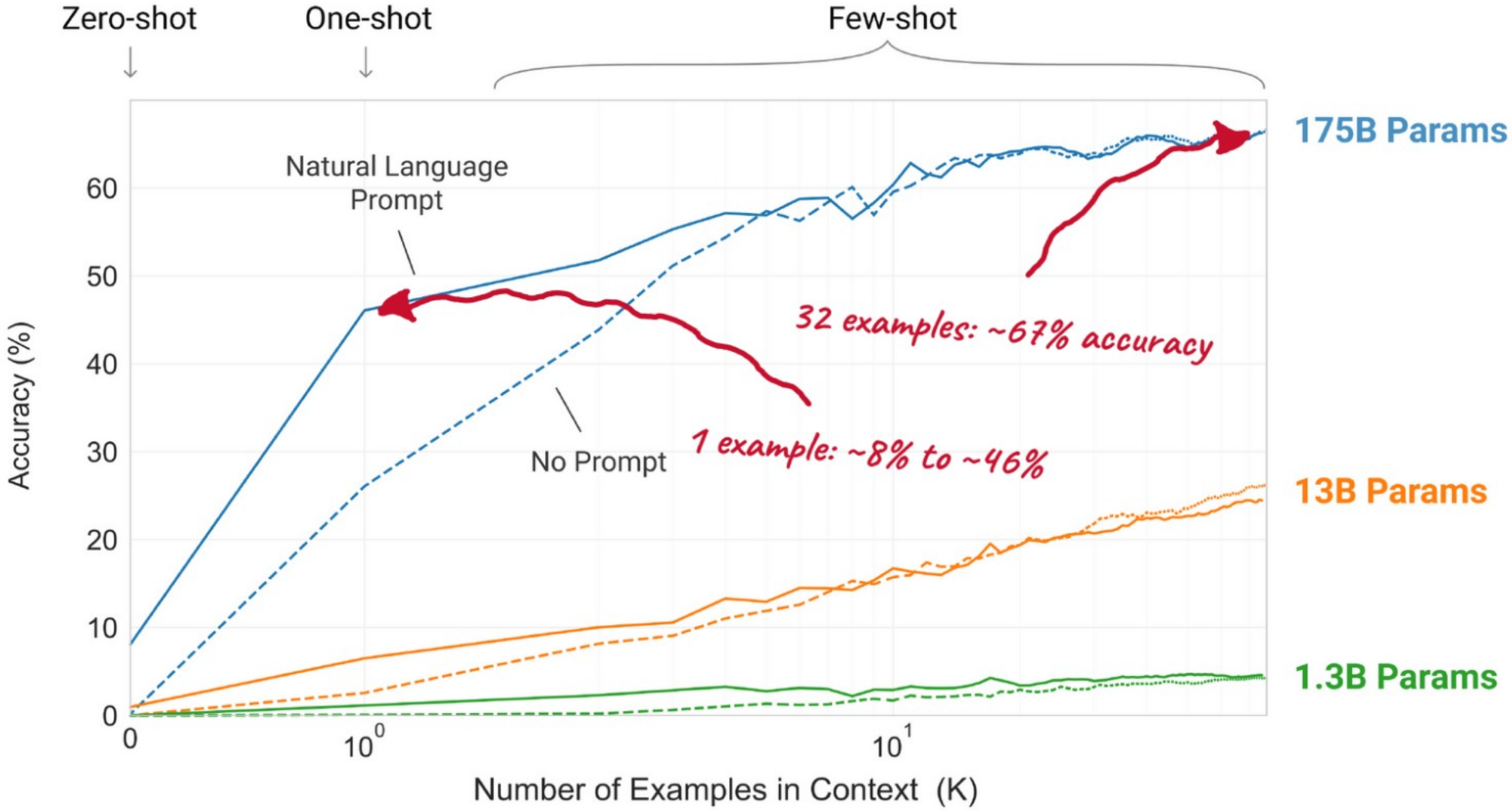
Quick overview

1. **Prompting methods:** in-context examples, reasoning prompts, self-consistency, format sensitivity
2. **Tool interfaces:** function schemas, tool descriptions, missing robot capabilities
3. **Harness design:** execution loops, action permissions, isolation, tool security, state and recovery
4. **Prompt optimization:** DSPy signatures, modules, bootstrapped examples, E2E results
5. **Observability:** traces, prompt versions, judge-score regression tests

1 THE CATALOG

What Still Works

In-Context Examples



Brown et al., "Language Models are Few-Shot Learners," 2020, arXiv:2005.14165.

What Stopped Mattering

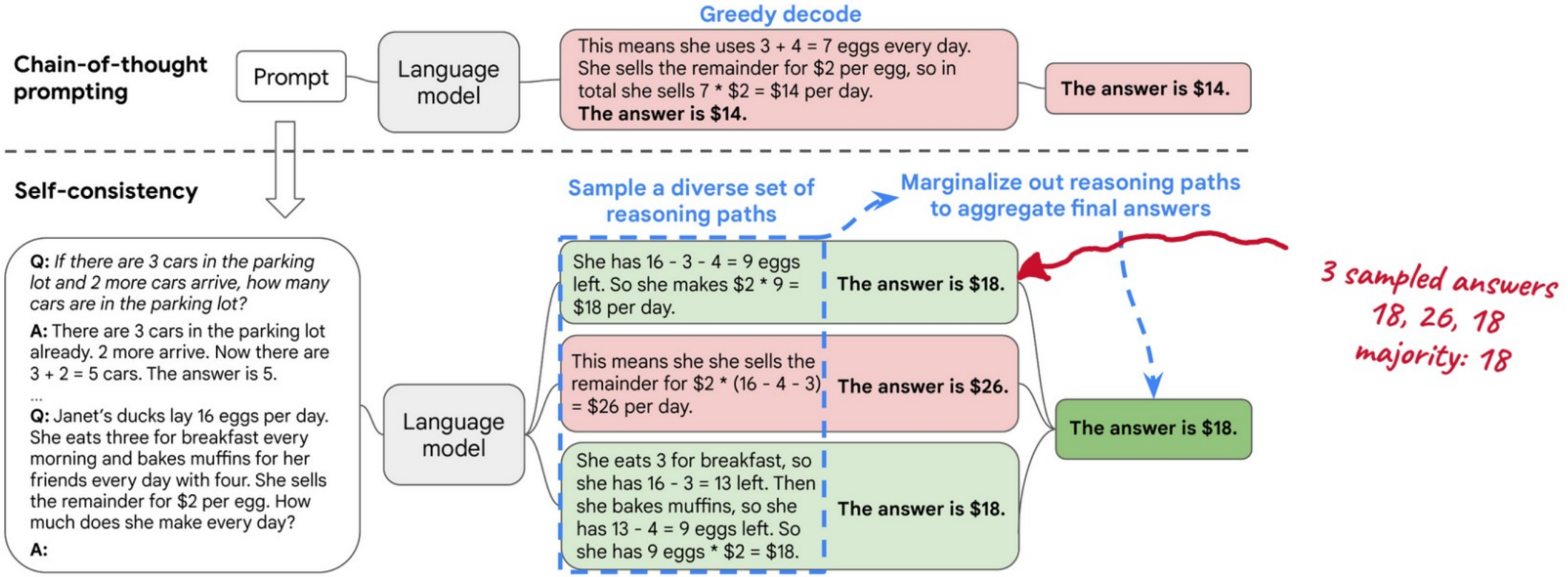
R1: few-shot accuracy decreases

- **Prompting Engineering:** When evaluating DeepSeek-R1, we observe that it is sensitive to prompts. Few-shot prompting consistently degrades its performance. Therefore, we recommend users directly describe the problem and specify the output format using a zero-shot setting for optimal results.

R1 recommendation:
zero-shot + output format

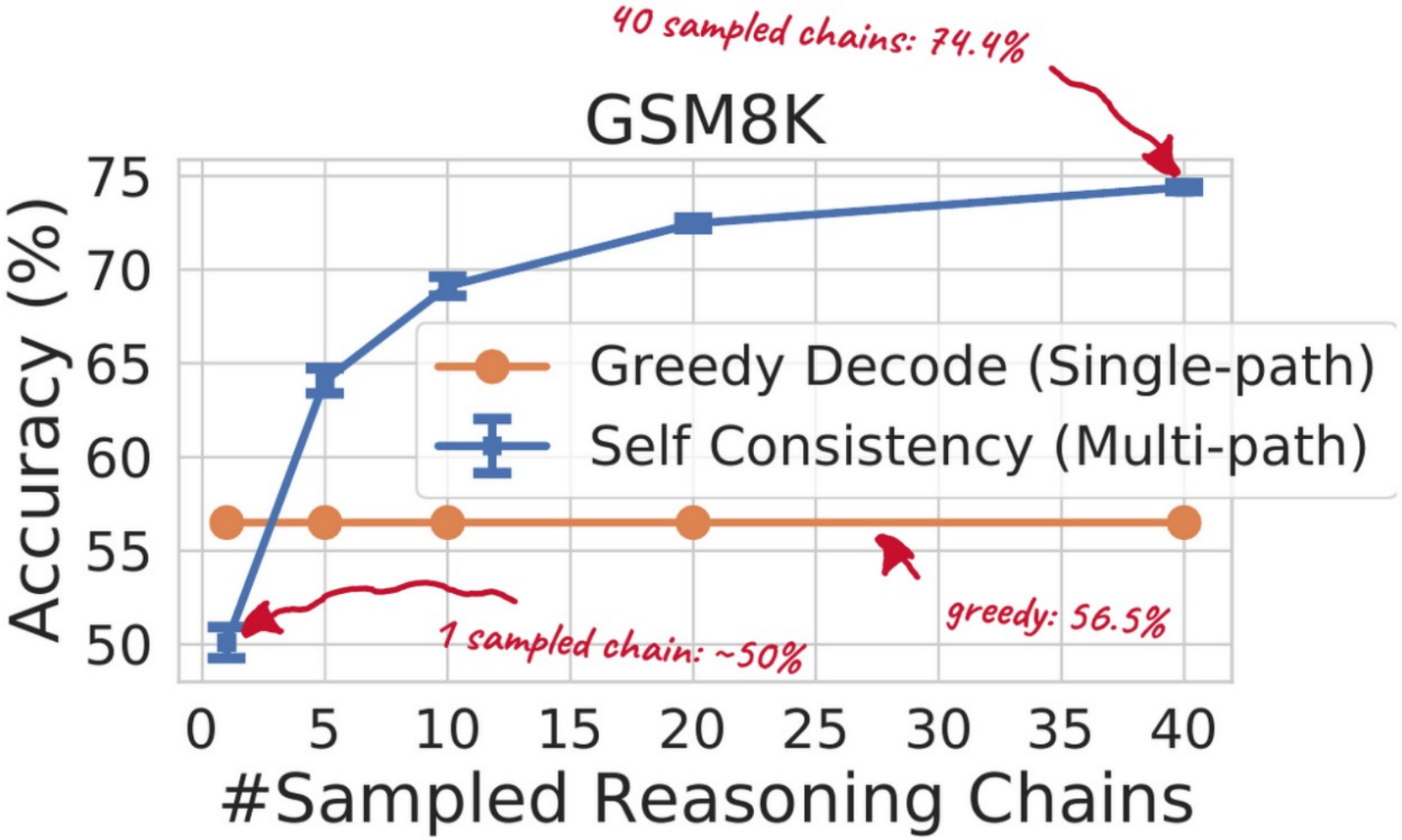
Kojima et al., "Large Language Models are Zero-Shot Reasoners," 2022, arXiv:2205.11916. DeepSeek-AI, "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning," 2025, arXiv:2501.12948.

Self-Consistency



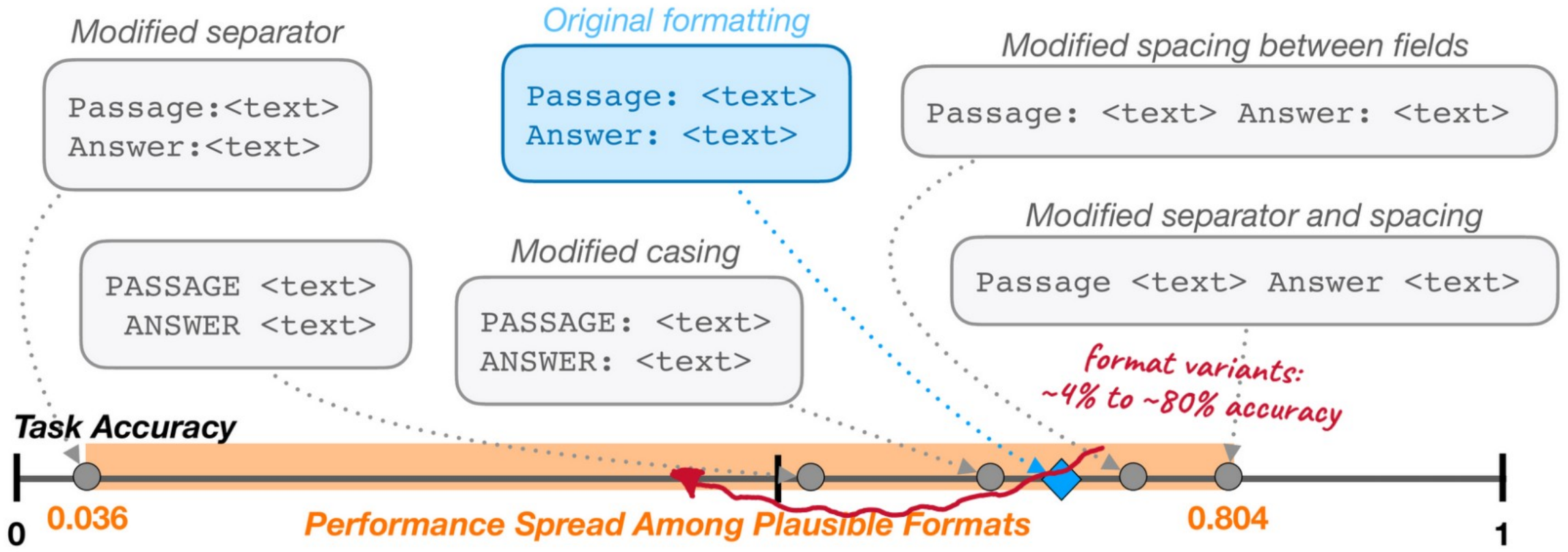
Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models," 2022, arXiv:2203.11171.

Self-Consistency



Wang et al., "Self-Consistency Improves Chain of Thought Reasoning in Language Models," 2022, arXiv:2203.11171.

Brittleness



Sclar et al., "Quantifying Language Models' Sensitivity to Spurious Features in Prompt Design," 2023, arXiv:2310.11324.

2 TOOL CALLING

The Missing Primitive

Function Schemas

```
tool       grade_model(run, corpus)
           run: one of pico, nano, micro, mini, base, base-tinystories
           corpus: one of shakespeare, tinystories, wikipedia, python-code
           description: "Score a saved Assignment 2 checkpoint on one evaluation corpus and
           return its teacher-forced loss in nats per character. `run` names the checkpoint: t
           ..." (778 characters)

user       how does my mini model do on python code

assistant  grade_model(run="mini", corpus="python-code")

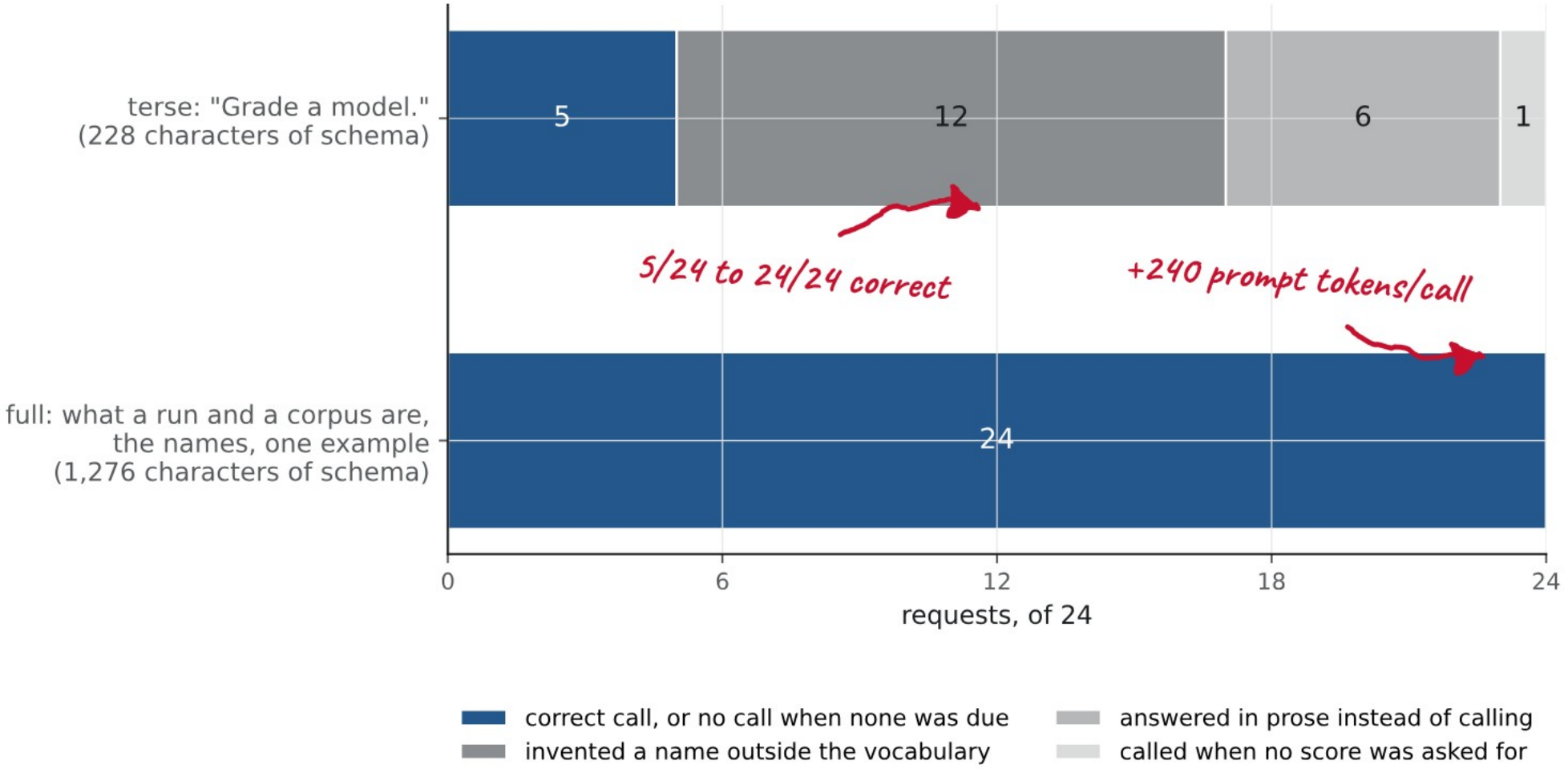
tool       {"nats_per_char": 1.43}

assistant  The mini model has a teacher-forced loss of 1.43 nats per character on the Python
           code evaluation corpus.
```



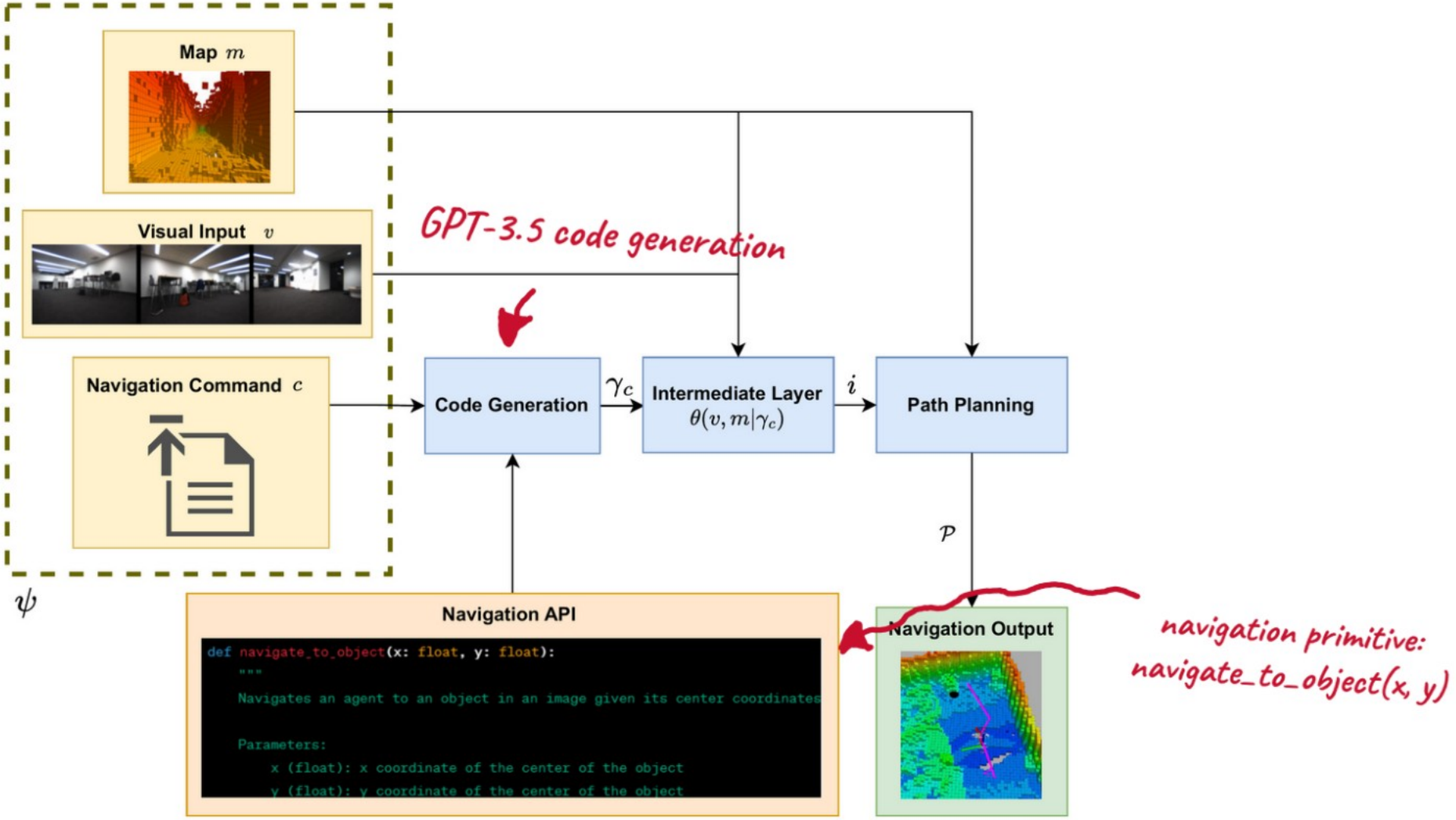
My run: qwen3 8B through Ollama on my RTX 4090, November 2.

Describing a Tool



My run: qwen3 8B through Ollama on my RTX 4090, 24 requests, three repeats each, November 2.

The Missing Primitive



Biggie, Mopidevi, Woods and Heckman, "Tell Me Where to Go: A Composable Framework for Context-Aware Embodied Robot Navigation," CoRL 2023, arXiv:2306.09523.

The Missing Primitive

"Go to the second floor"

```
floor_patches = ImagePatch(image).find('floor')
floor_patches.sort(key=lambda x: x.vertical_center)
if len(floor_patches) < 2:
    return {'function': 'None', 'error': 'Image does not contain at least
        two floors.'}
second_floor_patch = floor_patches[1]
```

second visible floor region

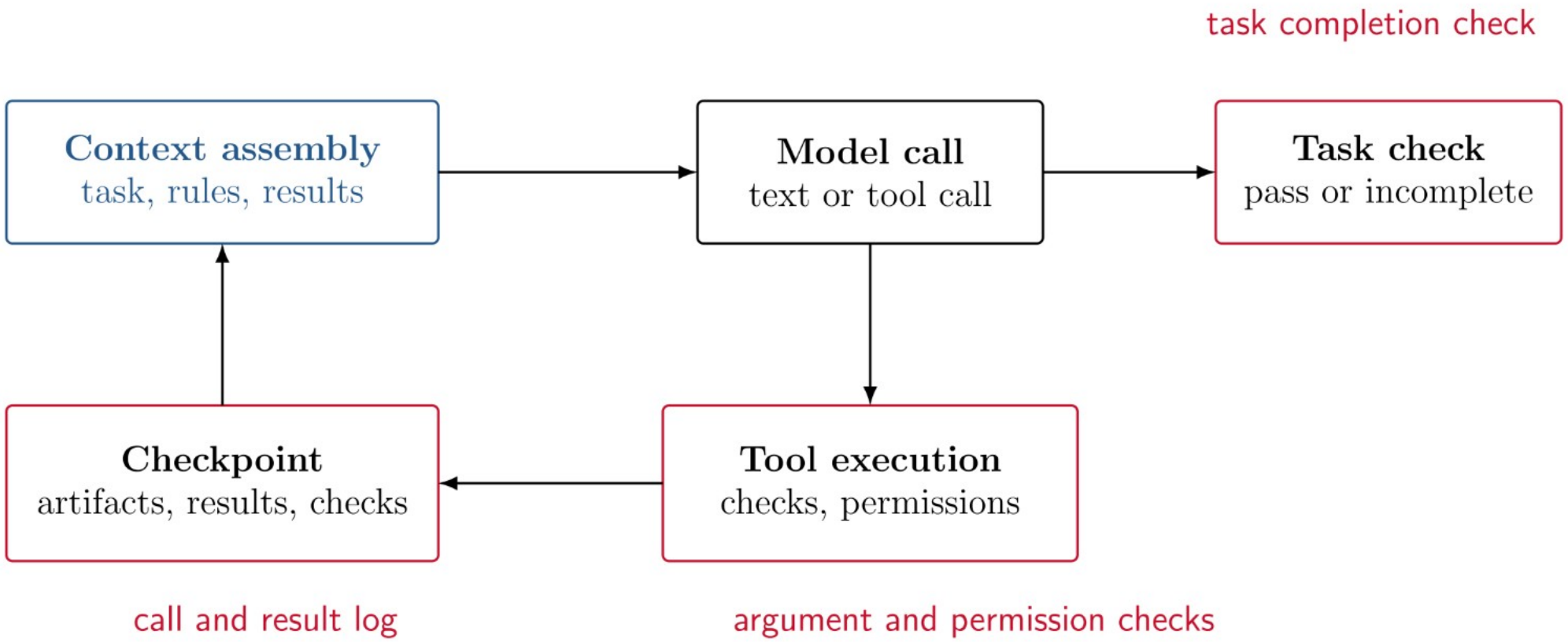
no action for unseen targets

Biggie, Mopidevi, Woods and Heckman, "Tell Me Where to Go: A Composable Framework for Context-Aware Embodied Robot Navigation," CoRL 2023, arXiv:2306.09523.

3 AGENT RUNTIME

Harness Design

Harness Components



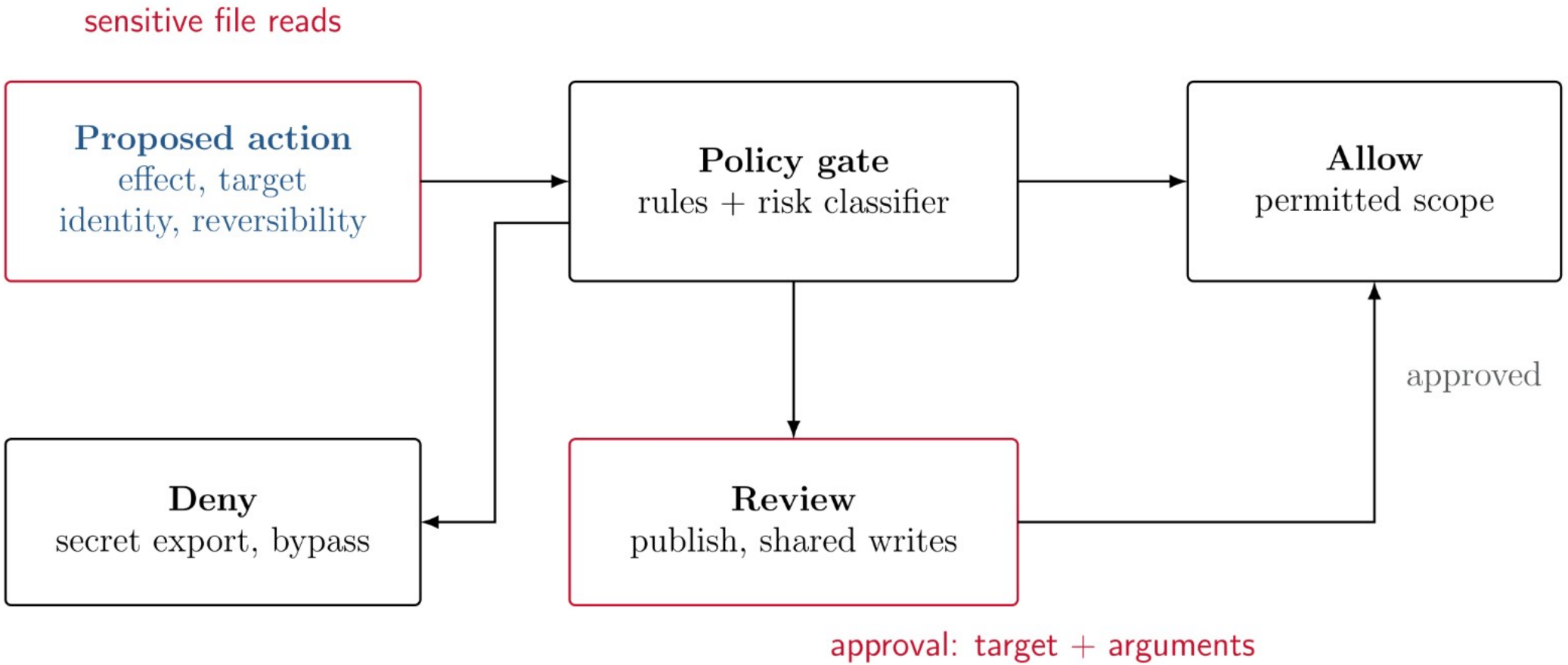
Anthropic, "Building Effective Agents," 2024.

Execution Loop

```
# Illustrative pseudocode: one tool call per turn
for turn in range(MAX_TURNS):
    reply = model(context(state), tools=SCHEMAS)
    if reply.is_final:
        return verify_or_reject(reply, state)
    call = validate(reply.tool_call, SCHEMAS)
    authorize(call, policy=POLICY, task=state.task)
    result = execute_isolated(call, timeout=TOOL_TIMEOUT)
    state.record(call.id, call, result)
    checkpoint(state)
return incomplete(state, reason="turn_limit")
```

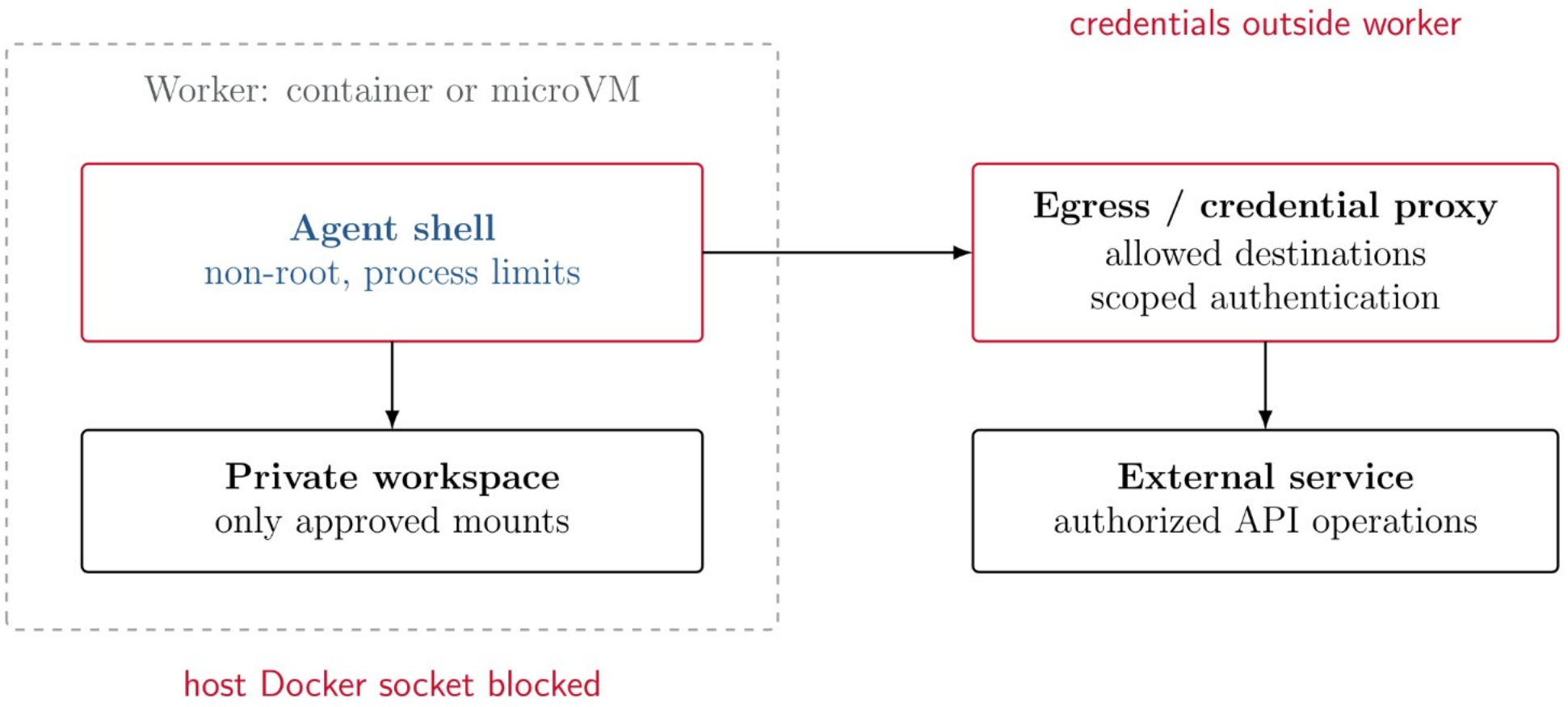
Anthropic, "Building Effective Agents," 2024.

Action Decisions



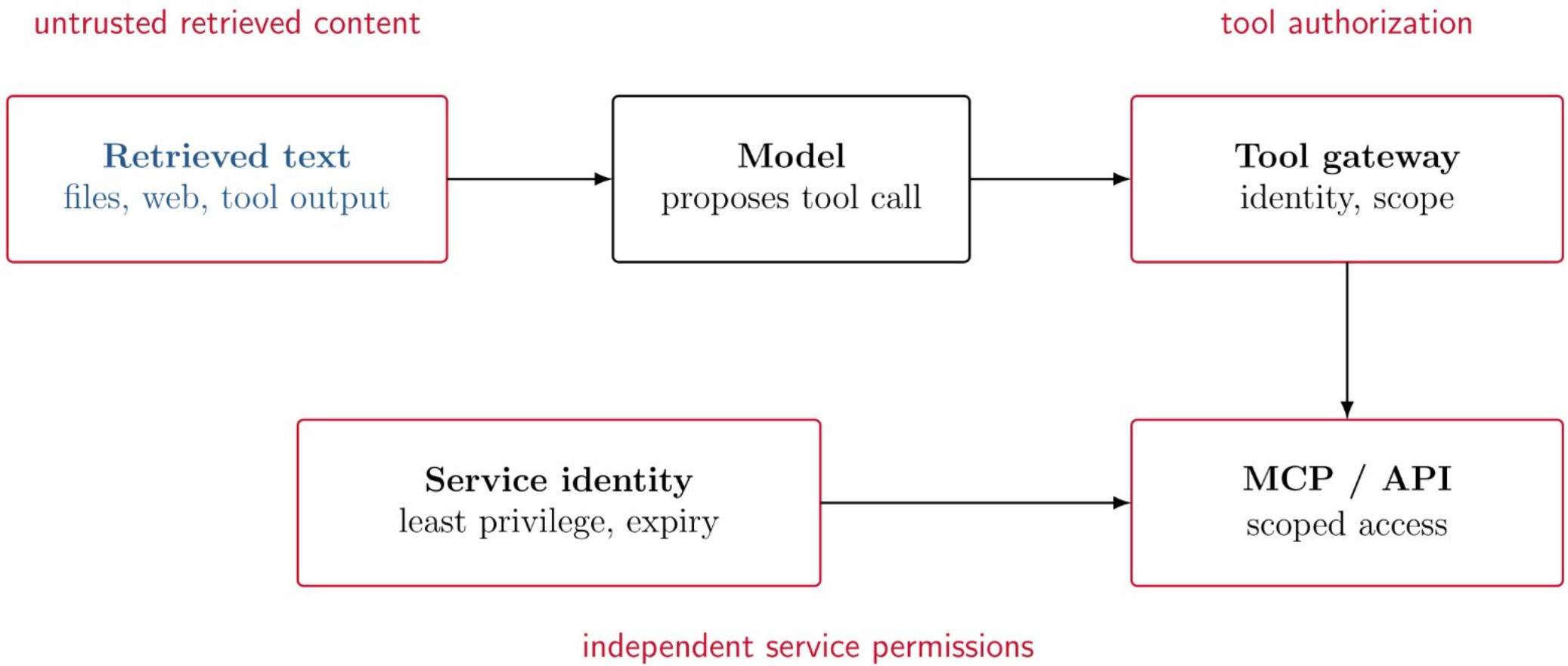
Illustrative policy, after Anthropic, "How We Built Claude Code Auto Mode," 2026.

Execution Isolation



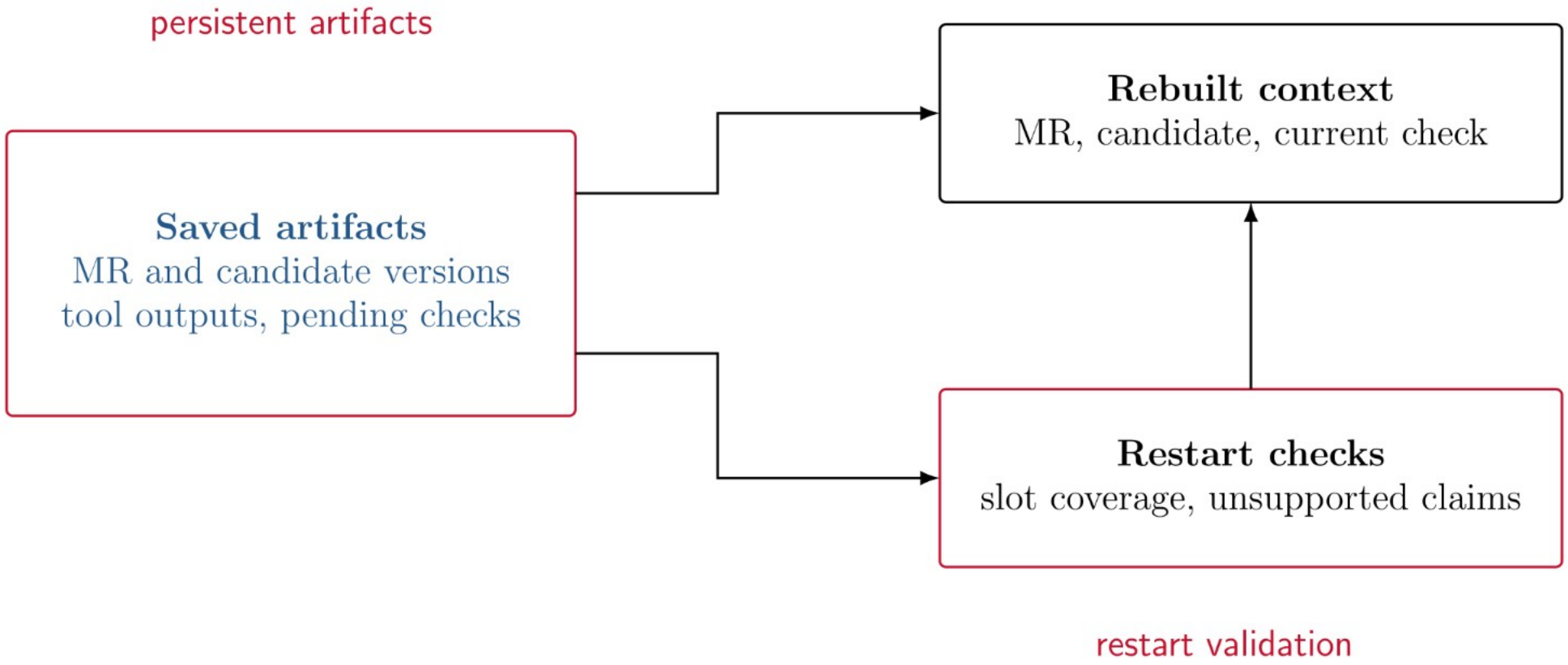
Schematic, after Anthropic, "Claude Code Sandboxing," 2025; Docker, "Engine Security" and "Sandbox Security," 2026.

Tool Trust Boundaries



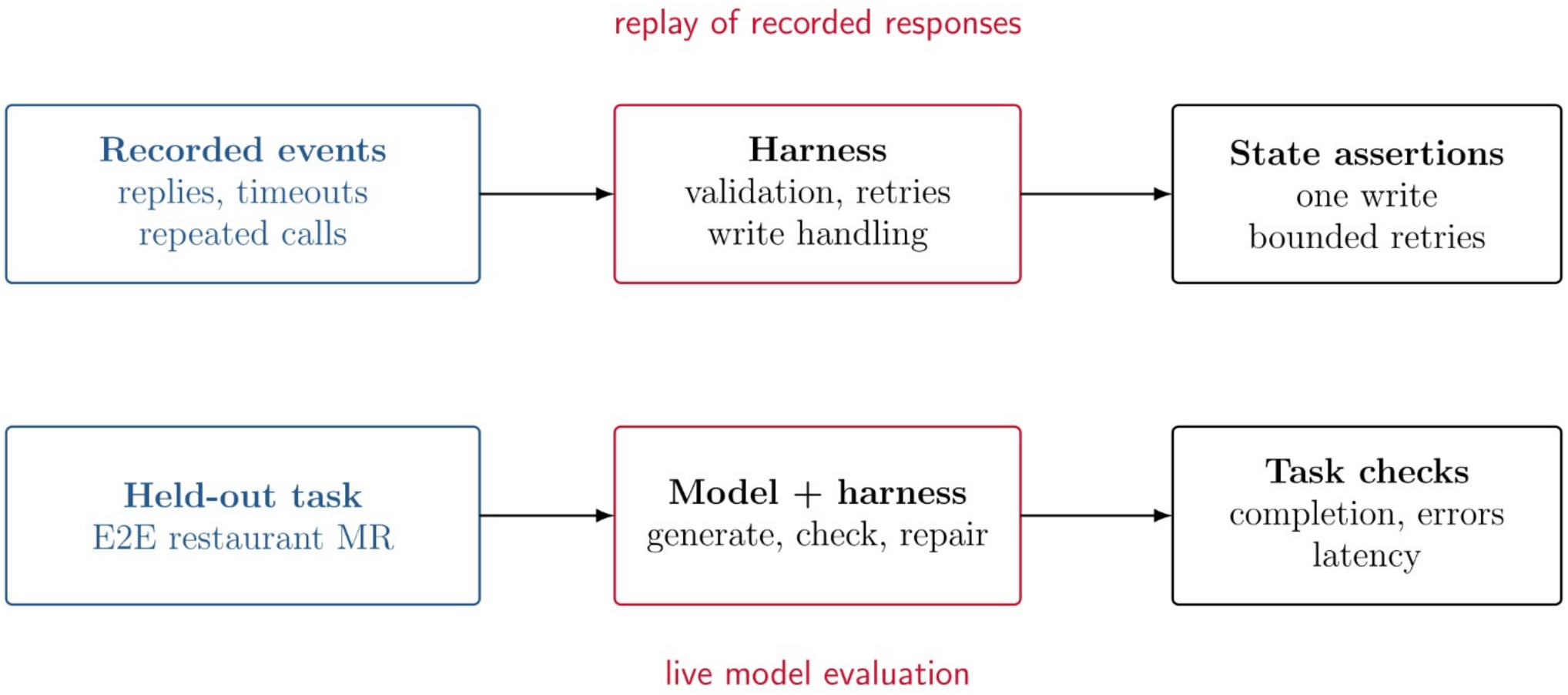
Schematic, after Claude Code, "Sandboxing" and "Security"; Docker, "Sandbox Architecture."

Context and State



Anthropic, "Effective Harnesses for Long-Running Agents," 2025.

Harness Tests



Illustrative replay cases for the course workflow.

4 PROMPT OPTIMIZATION

Compiling Prompts

Lengthy Strings

The ML community is rapidly exploring techniques for prompting language models (LMs) and for stacking them into pipelines that solve complex tasks. Unfortunately, existing LM pipelines are typically implemented using hard-coded "prompt templates", i.e. lengthy strings discovered via trial and error. Toward a more systematic approach for developing and optimizing LM pipelines, we introduce DSPy, a programming model that abstracts LM pipelines as *text transformation graphs*, i.e. imperative computation graphs where LMs are invoked through *declarative* modules. DSPy modules are *parameterized*, meaning they can learn (by creating and collecting demonstrations) how to apply compositions of prompting, finetuning, augmentation, and reasoning techniques. We design a compiler that will optimize any DSPy pipeline to maximize a given metric. We conduct

Khattab et al., "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," 2023, arXiv:2310.03714, abstract.

Signatures

```python
class RAG(dspy.Module):
    def __init__(self, num_passages=3):
        # 'Retrieve' will use the user's default retrieval settings unless overriden.
        self.retrieve = dspy.Retrieve(k=num_passages)
        # 'ChainOfThought' with signature that generates answers given retrieval & question.
        self.generate_answer = dspy.ChainOfThought("context, question -> answer")

    def forward(self, question):
        context = self.retrieve(question).passages
        return self.generate_answer(context=context, question=question)
```

ChainOfThought module
reasoning + answer

signature: inputs and output

Khattab et al., "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," 2023, arXiv:2310.03714.

The Optimizer

Program	Compilation	Training	GPT-3.5 Dev	GPT-3.5 Test	Llama2-13b-chat Dev	Llama2-13b-chat Test
vanilla	none	n/a	24.0	25.2	7.0	9.4
	fewshot	trainset	33.1	–	4.3	–
	bootstrap	trainset	44.0	–	28.0	–
	bootstrap×2	trainset	64.7	61.7	37.3	36.5
	+ensemble	trainset	62.7	61.9	39.0	34.6
CoT	none	n/a	50.0	–	26.7	–
	fewshot	trainset	63.0	–	27.3	–
	fewshot	+human_CoT	78.6	72.4	34.3	33.7
	bootstrap	trainset	80.3	72.9	43.3	–
	+ensemble	trainset	**88.3**	81.6	43.7	–
reflection	none	n/a	65.0	–	36.7	–
	fewshot	trainset	71.7	–	36.3	–
	bootstrap	trainset	83.0	76.0	44.3	40.2
	+ensemble	trainset	86.7	–	**49.0**	**46.9**

GPT-3.5:
25.2% to 81.6%

Llama 2 13B:
9.4% to 46.9%

Khattab et al., "DSPy: Compiling Declarative Language Model Calls into Self-Improving Pipelines," 2023, arXiv:2310.03714.

Compiled

```
system     Your input fields are:

           ...

user       name[Cotto], food[Chinese], customer rating[5 out of 5], f ...

assistant  Cotto is a Chinese restaurant with a 5 out of 5 customer r ...

user       name[The Eagle], eatType[coffee shop], food[Japanese], pri ...

assistant  The Eagle is a coffee shop serving Japanese food located i ...

           ... 6 more worked examples ...

user       name[Fitzbillies], eatType[coffee shop], food[Chinese], pr ...

model      Fitzbillies is a coffee shop serving Chinese food in the r ...
```

8 selected demonstrations

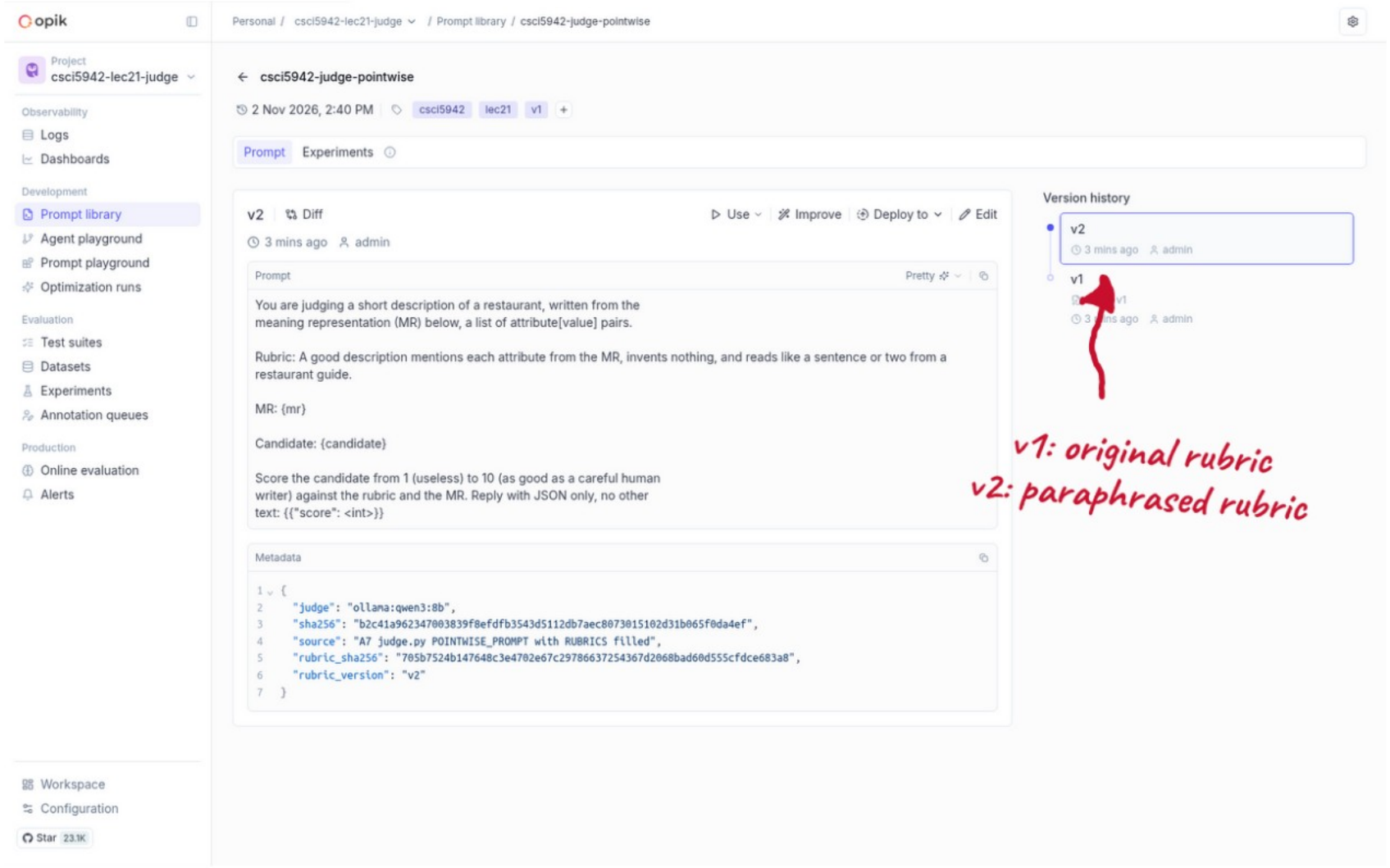
prompt tokens: 192 to 887
dev BLEU 55.1 to 58.0

My run: DSPy 3.4 with qwen3 8B through Ollama, 40 training and 60 held-out E2E meaning representations, November 2.

5 OBSERVABILITY

Prompts Under Test

Traces



My self-hosted Opik, the Assignment 7 judge through a local model, November 2.

A Regression Caught

```
$ python3 check_judge.py

judge         qwen3:8b (500a1f067a9f), n=60

prompt v1     c741c310d050  mean 8.62 +/- 0.28

prompt v2     b2c41a962347  mean 8.30 +/- 0.33

mean shift    -0.32 (budget +/-0.5)  ok

moved >= 2    4 of 60 = 7% (budget 10%)  ok

paired shift -0.317 +/- 0.189  (up 6, down 21, same 33)  FAIL

spearman      0.754

result        REGRESSION

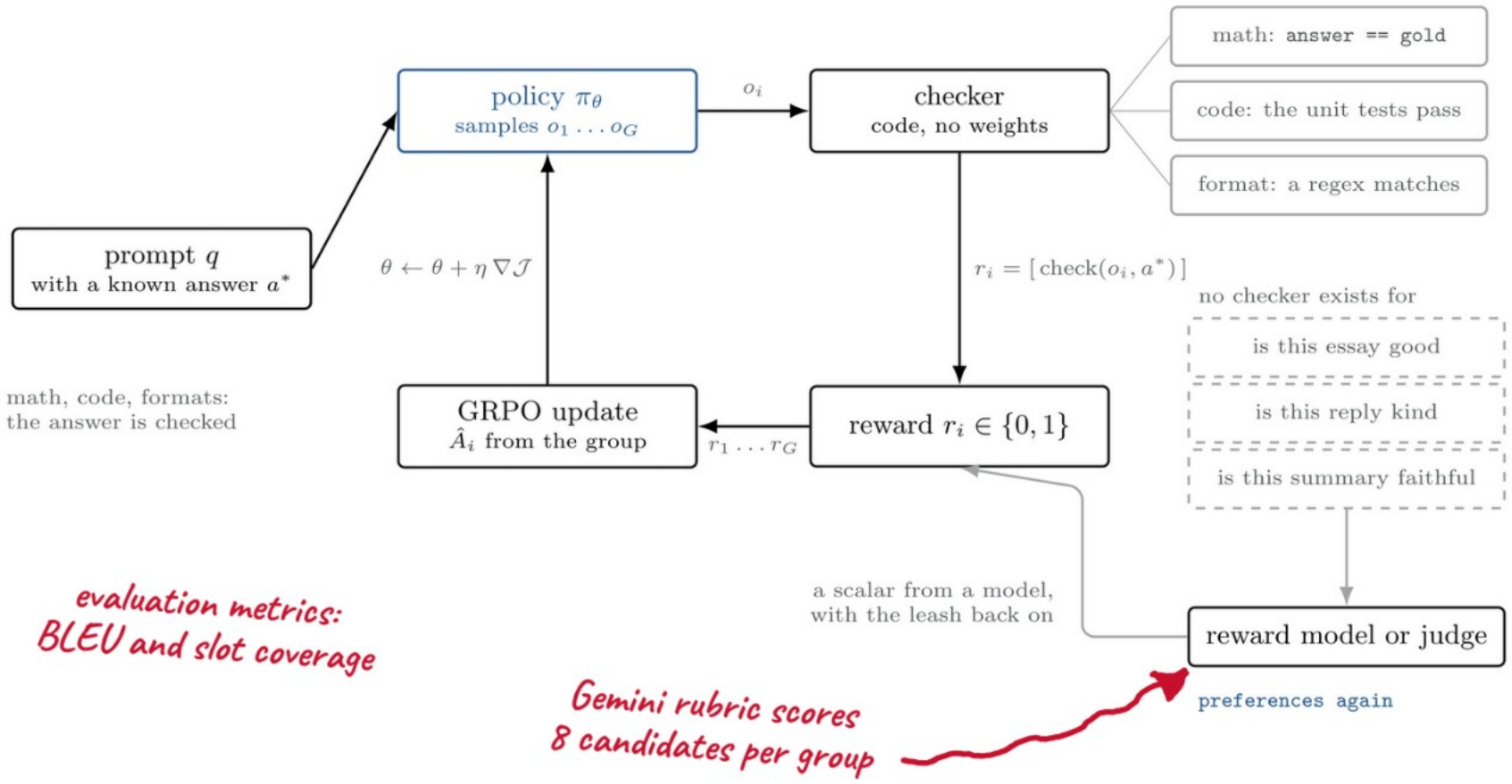
exit status 1
```



paired 95% CI excludes 0
check failed (exit 1)

My run of the Assignment 7 judge (qwen3 8B through Ollama) on 60 E2E test references, November 2.

Assignment 7



Schematic, after Bai et al., "Constitutional AI: Harmlessness from AI Feedback," 2022, arXiv:2212.08073.

Reading

Required

Khattab et al., DSPy · arXiv:2310.03714

Optional

Wang et al., Self-Consistency · arXiv:2203.11171

Anthropic, Building Effective Agents (2024)

Anthropic, Effective Harnesses for Long-Running Agents (2025)

Anthropic, Claude Code Auto Mode (2026); Docker, Sandbox Security