



AppWorld

A Controllable World of Apps and People
for Benchmarking Interactive Coding Agents

Harsh
Trivedi



ACL 2024

Bangkok, Thailand



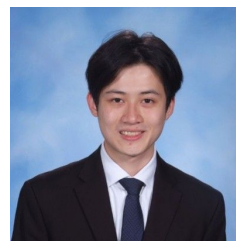
Tushar
Khot



Mareike
Hartmann



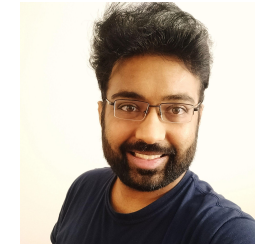
Ruskin
Manku



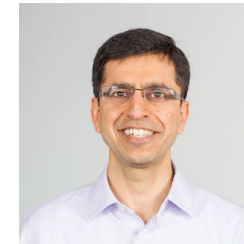
Vinty
Dong



Edward
Li



Shashank
Gupta



Ashish
Sabharwal



Niranjan
Balasubramanian



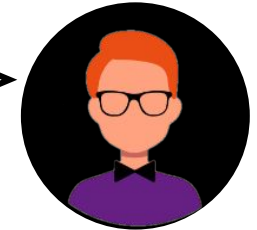
Stony Brook University



SAARLAND UNIVERSITY



Day-to-Day Tasks

Hey AI! Here are my app accounts. Do these tasks for me.



The last t-shirt I bought on  **Amazon**, doesn't fit me. Please initiate its return and buy it in one size larger.

I owe money to some of my friends on  **Splitwise**. Please pay them on  **Venmo** and clear Splitwise.


Play my  **Spotify** playlist with enough songs for the workout today. My workout plan is in  **SimpleNote**.

Can AI agents do such day-to-day tasks on our behalf?



Day-to-Day Tasks

Hey AI! Here are my app accounts. Do these tasks for me.



The last t-shirt I bought on  **Amazon**, doesn't fit me. Please initiate its return and buy it in one size larger.

I owe money to some of my friends on  **Splitwise**. Please pay them on  **Venmo** and clear Splitwise.

Play my  **Spotify** playlist with enough songs for the workout today. My workout plan is in  **SimpleNote**.



Can AI agents do such day-to-day tasks on our behalf?

Interactive Coding Agents for Personal Apps

LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Interactive Coding Agents for Personal Apps

LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.



Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



Interactive Coding Agents for Personal Apps

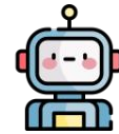
LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



Let me find Joe's today's workout plan.





```
token = simple_note.login(...)[“token”]  
notes = simple_note.search_notes(“workout”, token)  
print(notes[0]) # found one, show it ...
```

APIs

Interactive Coding Agents for Personal Apps

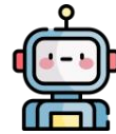
LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



Let me find Joe's today's workout plan.





```
token = simple_note.login(...)[“token”]  
notes = simple_note.search_notes(“workout”, token)  
print(notes[0]) # found one, show it ...
```

... Monday: ... 25 mins ... Tuesday ... 45 mins ...

Interactive Coding Agents for Personal Apps

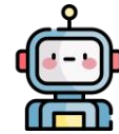
LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



Let me find Joe's today's workout plan.



```
token = simple_note.login(...)[“token”]  
notes = simple_note.search_notes(“workout”, token)  
print(notes[0]) # found one, show it ...
```

... Monday: ... 25 mins ... Tuesday ... 45 mins ...

Plan is day-wise, let's see what day is today.





```
print(datetime.now().strftime(“%A”))
```

Tuesday

Environment Interaction

Interactive Coding Agents for Personal Apps

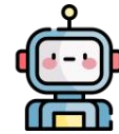
LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



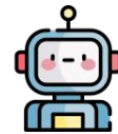
Let me find Joe's today's workout plan.



```
token = simple_note.login(...)[“token”]  
notes = simple_note.search_notes(“workout”, token)  
print(notes[0]) # found one, show it ...
```

... Monday: ... 25 mins ... Tuesday ... 45 mins ...

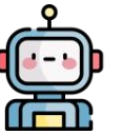
Plan is day-wise, let's see what day is today.



```
print(datetime.now().strftime(“%A”))
```

Tuesday



Today is Tuesday, so it's 45 mins.
Now, let me play Joe's playlist with
enough songs for this duration.



Environment Interaction

Interactive Coding Agents for Personal Apps

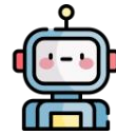
LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Joe



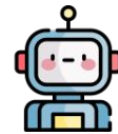
Let me find Joe's today's workout plan.



```
token = simple_note.login(...)[“token”]  
notes = simple_note.search_notes(“workout”, token)  
print(notes[0]) # found one, show it ...
```

... Monday: ... 25 mins ... Tuesday ... 45 mins ...

Plan is day-wise, let's see what day is today.

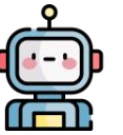


```
print(datetime.now().strftime(“%A”))
```

Tuesday

Environment Interaction

Today is Tuesday, so it's 45 mins.
Now, let me play Joe's playlist with enough songs for this duration.





```
workout_mins = 45  
token = spotify.login(...)[“token”]  
playlists = spotify.playlist_library(token)  
for playlist in playlists:  
    duration = 0  
    for id in playlist[“song_ids”]:  
        duration += spotify.song(id)[“duration”]  
    if total_duration >= workout_mins:  
        spotify.play(playlist[“id”], token)  
        break
```

Rich Code

10

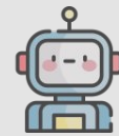
Interactive Coding Agents for Personal Apps

LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.



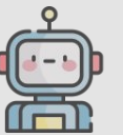
Let me find Joe's today's workout plan.



Today is Tuesday, so it's 45 mins.

```
token = simple_note_api.get_token()
notes = simple_note_api.get_notes(token)
print(notes[0])
```

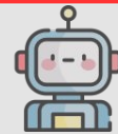
st with
on.



Seemingly simple day-to-day tasks
can be quite complex, with
Two Key Requirements 

... Monday: ...

Plan is day-wise, let's see what day is today.



```
print(datetime.now().strftime("%A"))
```

```
for playlist in playlists:
    duration = 0
    for id in playlist["song_ids"]:
        duration += spotify.song(id)["duration"]
    if total_duration >= workout_mins:
        spotify.play(playlist["id"], token)
        break
```

Tuesday

Environment Interaction

Rich Code

Interactive Coding Agents for Personal Apps

LMs could tackle such tasks by calling **APIs** with **rich** & **interactive coding**.

Play my Spotify playlist with enough songs for the workout today



How can we **develop & benchmark** such rich and interactive coding **agents for complex digital tasks** in a **rigorous & reproducible** manner?

```
print(datetime.now().strftime("%A"))
```

Tuesday

Environment Interaction

```
duration += spotify.song(id)["duration"]  
if total_duration >= workout_mins:  
    spotify.play(playlist["id"], token)  
    break
```

Rich Code

12

How to Benchmark such Interactive Coding Agents?

We need three ingredients

How to Benchmark such Interactive Coding Agents?

We need three ingredients

Environment



A rich & reproducible execution environment of many API-operable apps

How to Benchmark such Interactive Coding Agents?

We need three ingredients

Environment



A rich & reproducible execution environment of many API-operable apps

Benchmark



A set of **complex tasks** needing API calls with **rich & interactive coding**

How to Benchmark such Interactive Coding Agents?

We need three ingredients

Environment



A rich & reproducible execution environment of many API-operable apps

Benchmark



A set of **complex tasks** needing API calls with **rich & interactive coding**

Evaluation Framework



A **robust & programmatic** evaluation framework for checking goal completion

How to Benchmark such Interactive Coding Agents?

We need three ingredients

Past works don't support them

Environment

 A rich & reproducible execution environment of many API-operable apps

Not rich & reproducible

Small num. API implementations or unstable real APIs.


Benchmark

 A set of complex tasks needing API calls with rich & interactive coding

Not rich & interactive coding

Tasks require a simple sequence of 1-4 API calls.

Evaluation Framework

 A robust & programmatic evaluation framework for checking goal completion

Not robust & programmatic
human/LM judge or reference -based evaluation (task have many solutions!)

Gorilla (Patil et. al), **ToolBench** (Qin et. al), **API-Bank** (Li et. al), **ToolTalk** (Farn et. al), **RestBench** (Song et. al) ... 17

How to Benchmark such Interactive Coding Agents?



AppWorld


Environment

 A rich & reproducible execution environment of many API-operable apps

Benchmark

 A set of **complex tasks** needing API calls with **rich & interactive coding**

Evaluation Framework

 A **robust & programmatic** evaluation framework for checking goal completion

Engine

Benchmark

Evaluation

Engine

Benchmark

Evaluation

Amazon Todoist Venmo

SimpleNote Spotify Splitwise

Phone Gmail FileSystem

Fully Controllable
Local DB & API Server

Apps

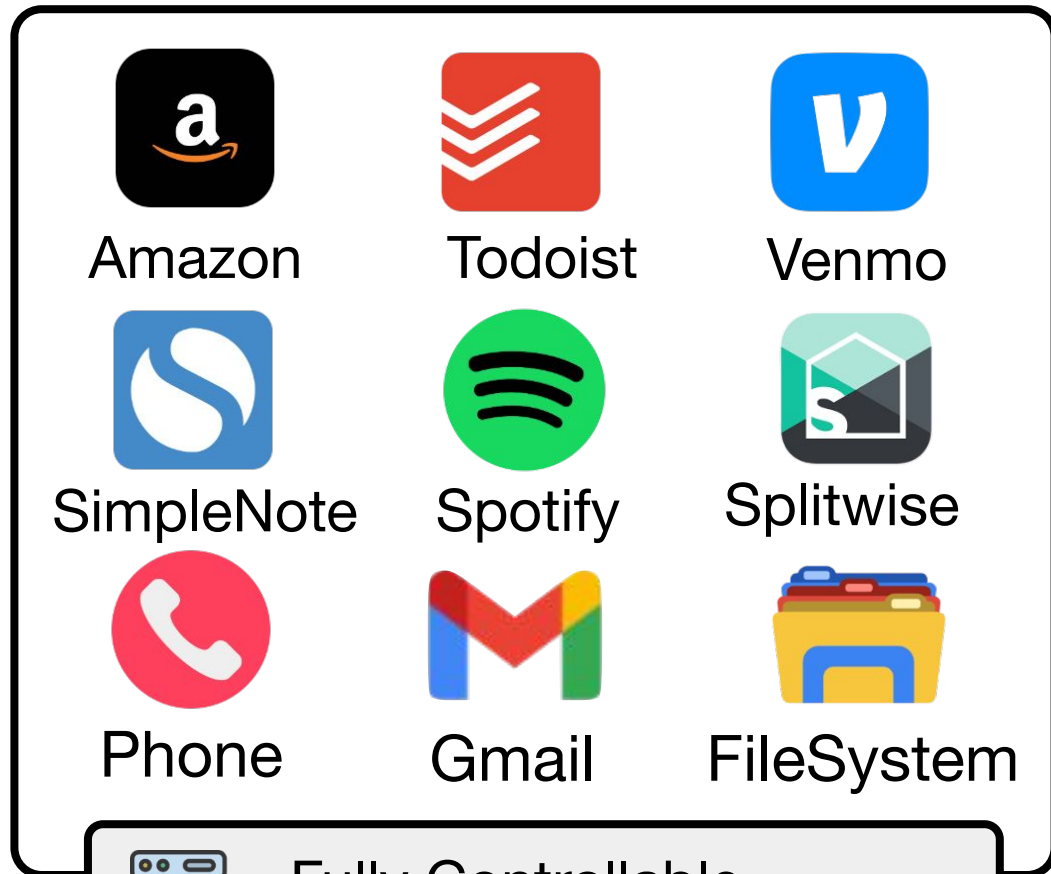
Has implementation of many day-to-day apps in a local backend of APIs and databases.

Provides full **control** over its DB **state** enabling challenging task construction & robust evaluation.

Engine

Benchmark

Evaluation



A grid of nine application icons arranged in three rows and three columns. The icons are: Amazon (black square with white 'a'), Todoist (red square with white lines), Venmo (blue square with white 'v'), SimpleNote (blue square with white 'S'), Spotify (green circle with black lines), Splitwise (teal square with white 'S'), Phone (red circle with white phone receiver), Gmail (colorful 'M'), and FileSystem (yellow folder icon).

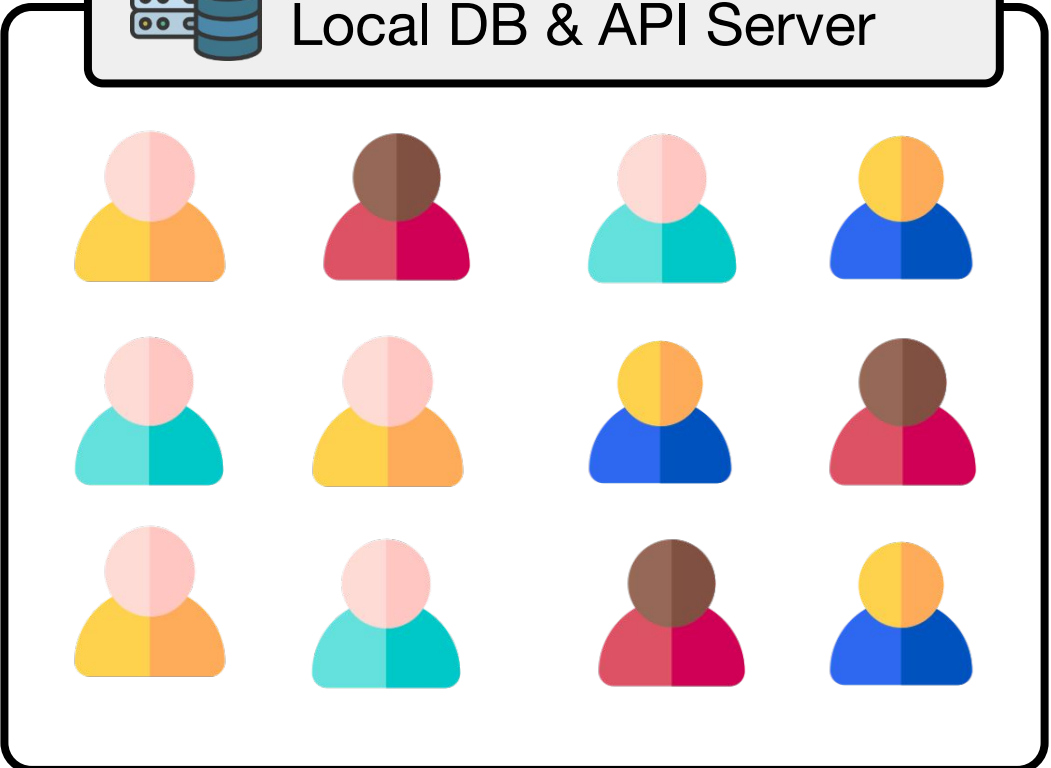
Apps

Has implementation of many day-to-day apps in a local backend of APIs and databases.

Provides full **control** over its DB **state** enabling challenging task construction & robust evaluation



Fully Controllable
Local DB & API Server



A grid of 12 human avatars arranged in three rows and four columns. Each avatar is a stylized head and shoulders figure with a different skin tone and hair color, representing a diverse population of users.

People

Apps populated with realistic simulated digital activities of people on their app accounts.

Provides a **realistic environment foundation** to build tasks on top of

Engine

Benchmark

Evaluation



Amazon



Todoist



Venmo



SimpleNote



Spotify



Splitwise



Phone



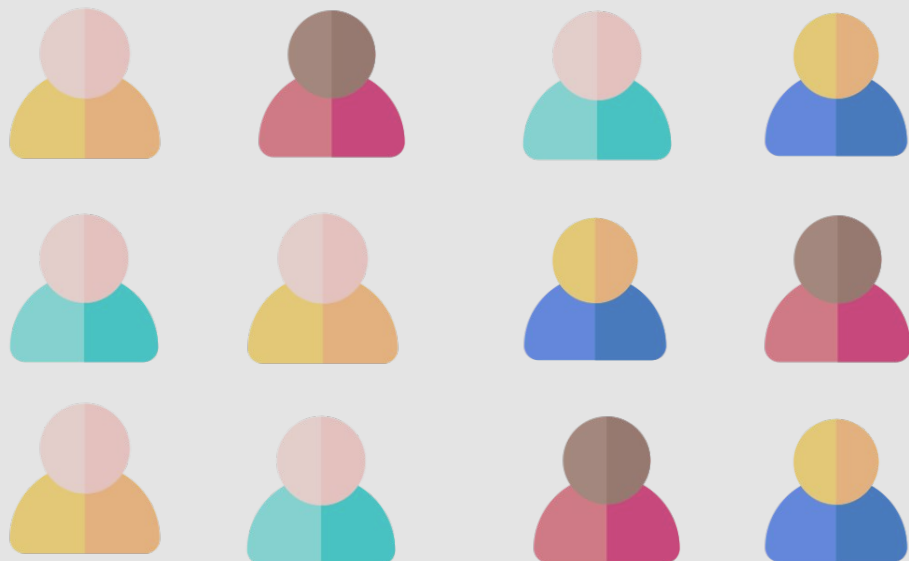
Gmail



FileSystem



Fully Controllable
Local DB & API Server



Apps

Has implementation of many day-to-day apps in a local backend of APIs and databases.

- ⇒ **High Fidelity:** 60K+ code lines, 457 APIs, 100+ DB tables
- ⇒ **Reliable:** 1700+ unit tests
- ⇒ **Documented API specs**

People

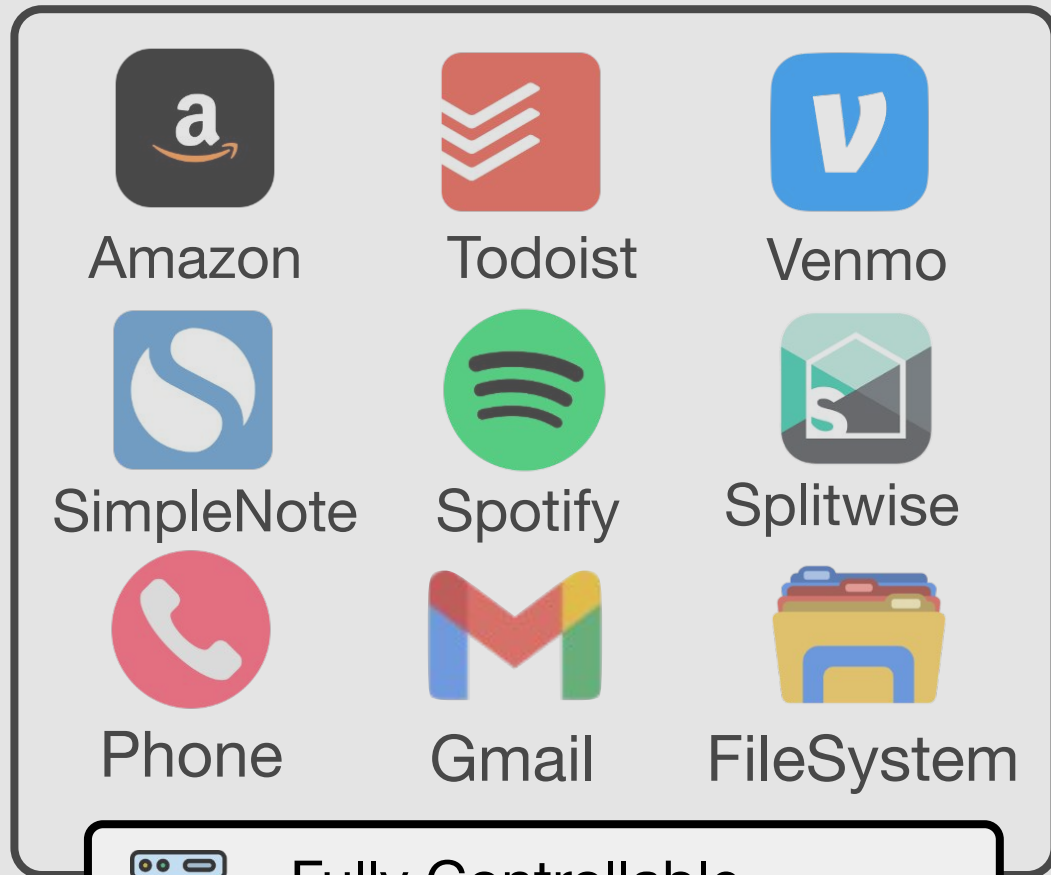
Apps populated with realistic simulated digital activities of people on their app accounts.

Provides a **realistic environment foundation** to build tasks on top of

Engine

Benchmark

Evaluation



Apps

Has implementation of many day-to-day apps in a local backend of APIs and databases.

- ⇒ **High Fidelity:** 60K+ code lines, 457 APIs, 100+ DB tables
- ⇒ **Reliable:** 1700+ unit tests
- ⇒ **Documented API specs**

People

Apps populated with realistic simulated digital activities of people on their app accounts.

- ⇒ 106 fictitious **people**
- ⇒ Related via various **relationships**
- ⇒ Realistic (inter)personal **digital activities**



Fully Controllable
Local DB & API Server

[phone.message]

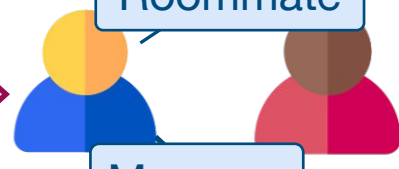
Hey Roomie, have you seen my car keys?



Roommate

[amazon.order]

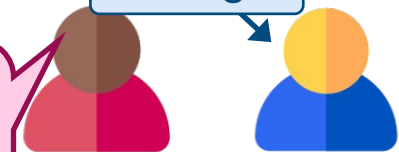
Ordered two red T-shirts for home delivery.



Manager

[gmail.send_email]

Can you approve my leave for tomorrow?



Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

Naturally require
rich & interacting
coding

Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

3x

ground

Task



**Generator
Code**

Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

3x

ground

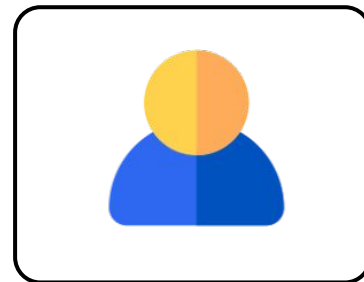
Task



**Generator
Code**

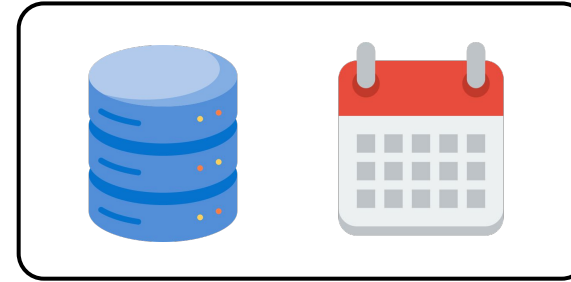
Play my Spotify playlist with ...

Instruction



Supervisor

Person in AppWorld
assigning the task



World State

The Initial DB State &
Time Now in World

Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

3x

ground

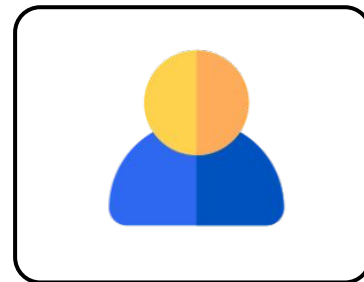
Task



Generator Code

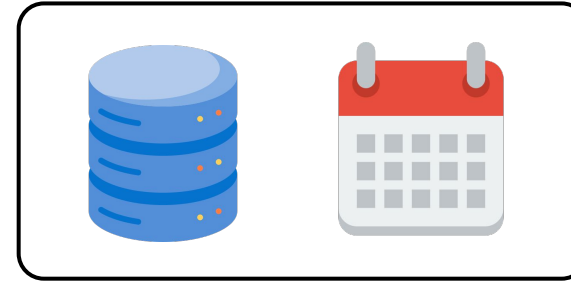
Play my Spotify playlist with ...

Instruction



Supervisor

Person in AppWorld
assigning the task



World State

The Initial DB State &
Time Now in World

**Solution Code
+ Eval. Tests**



To Verify
Solvability



Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

3x

ground

Task

Play my Spotify playlist with ...

We write them *individually*
for every task/scenario
(40K+ lines of code)



Generator
Code

P

assigning the task Time Now in World

&

Solution Code
+ Eval. Tests



To Verify
Solvability



Engine

Benchmark

Evaluation

Day-to-day rich & interactive coding tasks developed on Engine

Play Spotify ..
with enough
songs ...

Scenario

3x

ground

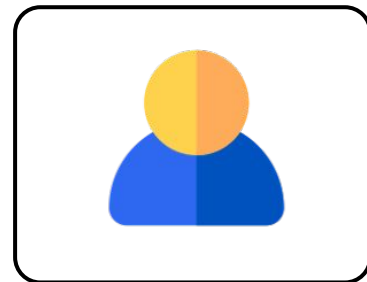
Task



Generator
Code

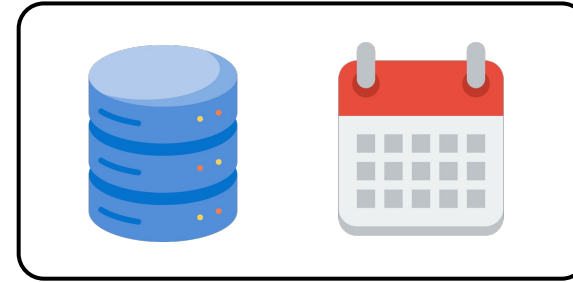
Play my Spotify playlist with ...

Instruction



Supervisor

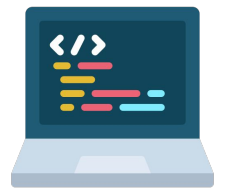
Person in AppWorld
assigning the task



World State

The Initial DB State &
Time Now in World

Solution Code
+ Eval. Tests



To Verify
Solvability



Agent
Given



Task
Instruction



Supervisor's
App Credentials



API Docs
of all Apps



Interactive
Coding Shell
(w/ API client)



Engine

Benchmark

Evaluation



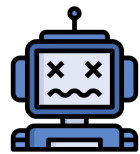
Generator Code for each scenario ensures four task properties

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Property



Is Well-Defined



Has Distractors



Has Hurdles



Has Variations

Purpose

To be solvable

To necessitate thorough reasoning
(avoid solvability by shortcuts)

To assess reliability across different
instruction / state variations.



Engine

Benchmark

Evaluation



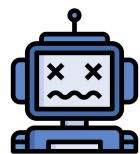
Generator Code for each scenario ensures four task properties

Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Property



Is Well-Defined



Has Distractors



Has Hurdles



Has Variations

Purpose

To be solvable

Supervisor has several playlists.
Diff. workout durations for diff. days
Only one playlist qualifies for today

Skip a reasoning step ⇒ **Fail!**

To assess reliability across different instruction / state variations.



Engine

Benchmark

Evaluation



Generator Code for each scenario ensures four task properties

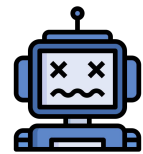
Play my  **Spotify** playlist with enough songs for the workout today.
My workout plan is in  **SimpleNote**.

Property

Purpose



Is Well-Defined



Has Distractors



Has Hurdles



Has Variations

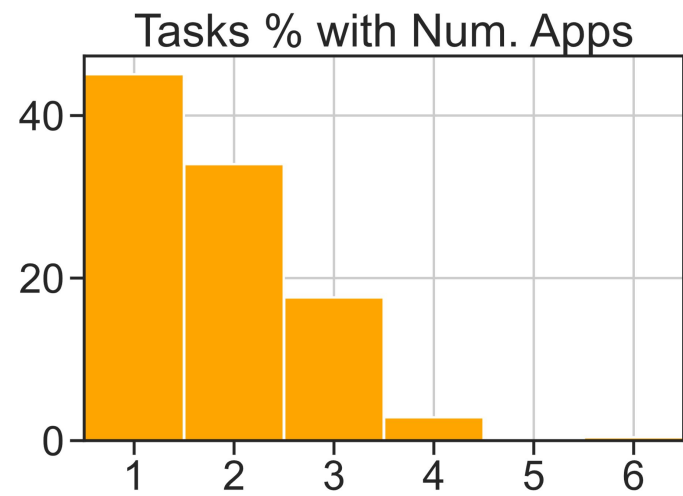
To be solvable

Supervisor has several playlists.
Diff. workout durations for diff. days
Only one playlist qualifies for today

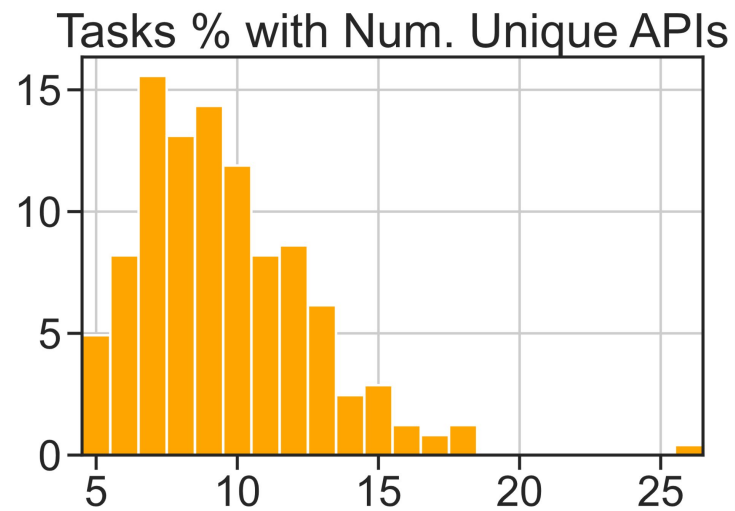
Skip a reasoning step ⇒ **Fail!**

Careful task construction possible as
Engine gives **full control over its state**

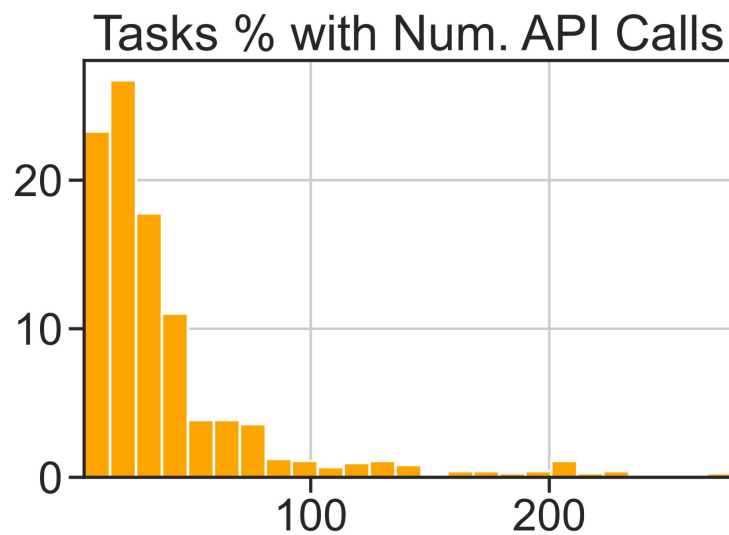
Dataset Statistics



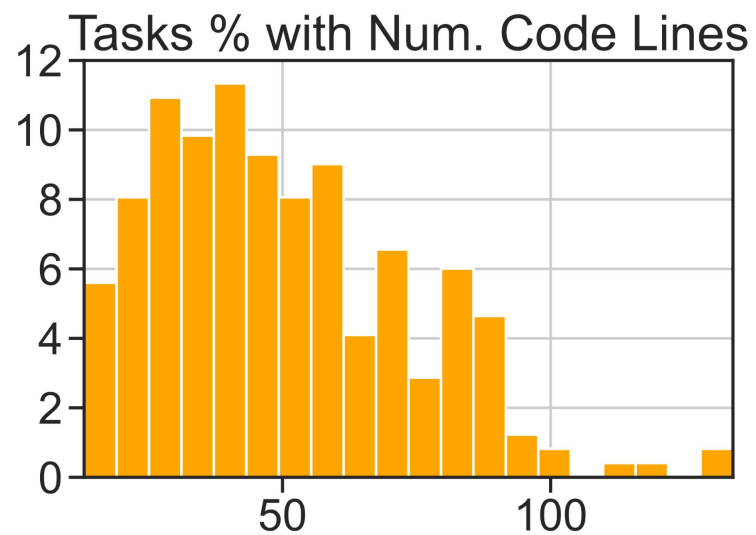
1-4 apps



5-25 Unique APIs



10-200 API Calls



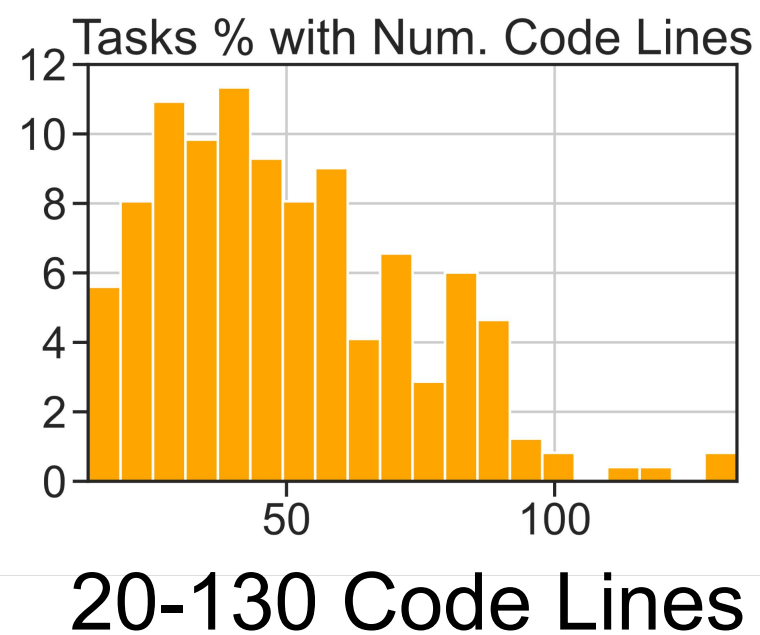
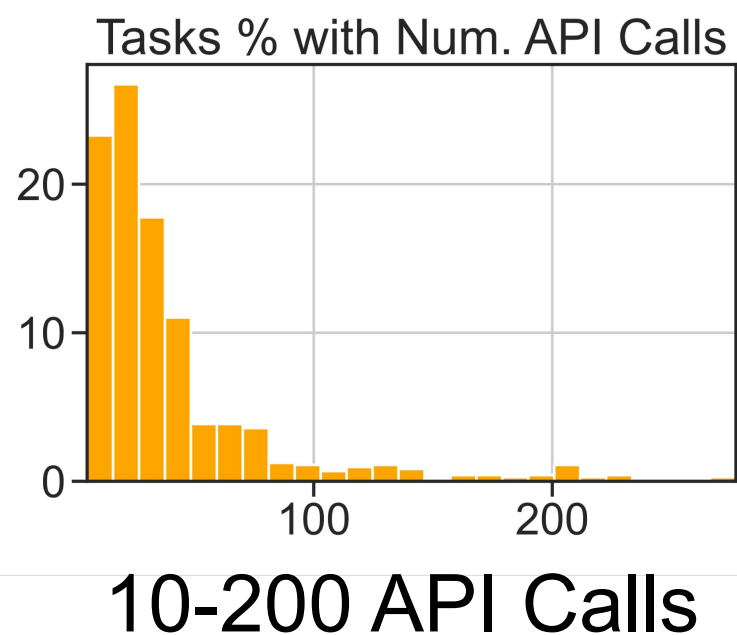
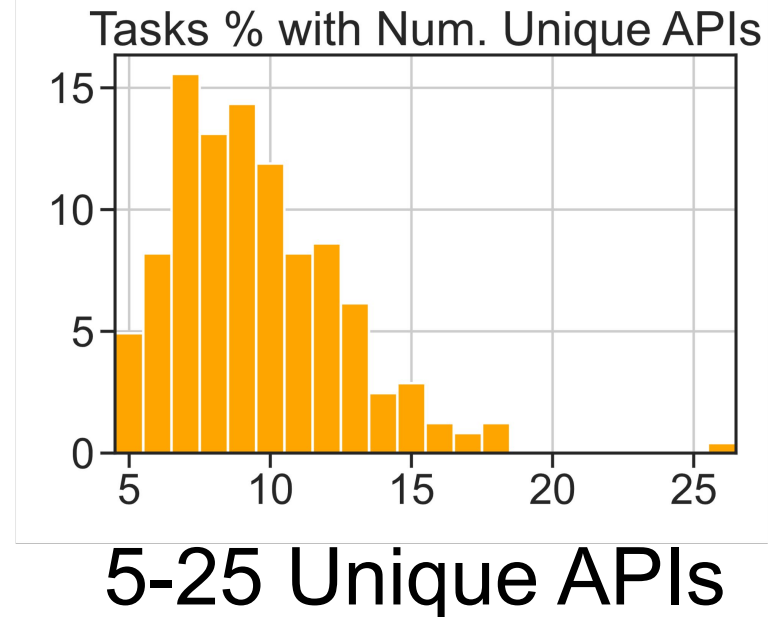
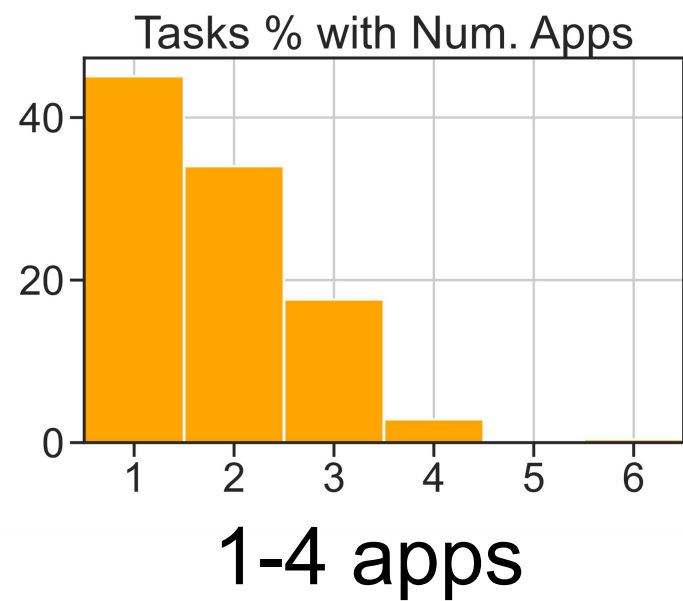
20-130 Code Lines

Engine

Benchmark

Evaluation

Dataset Statistics



Splits: 750
⇒ Train+Dev: 165
⇒ Test Normal: 168
⇒ **Test Challenge:** 417

Main Test Set
Challenging +
Unseen Apps

Engine

Benchmark

Evaluation

How to robustly evaluate agents on such tasks?

Engine

Benchmark

Evaluation

How to robustly evaluate agents on such tasks?

Many valid ways of completing the goal.

E.g., Amazon receipt downloadable from its API or email confirmation.

Many ways of causing collateral damage.

E.g., deleting emails when asked for placing an Amazon order.

Engine

Benchmark

Evaluation

How to robustly evaluate agents on such tasks?

Many valid ways of completing the goal.

E.g., Amazon receipt downloadable from its API or email confirmation.

Many ways of causing collateral damage.

E.g., deleting emails when asked for placing an Amazon order.



Comparison to a reference code/API calls isn't suitable

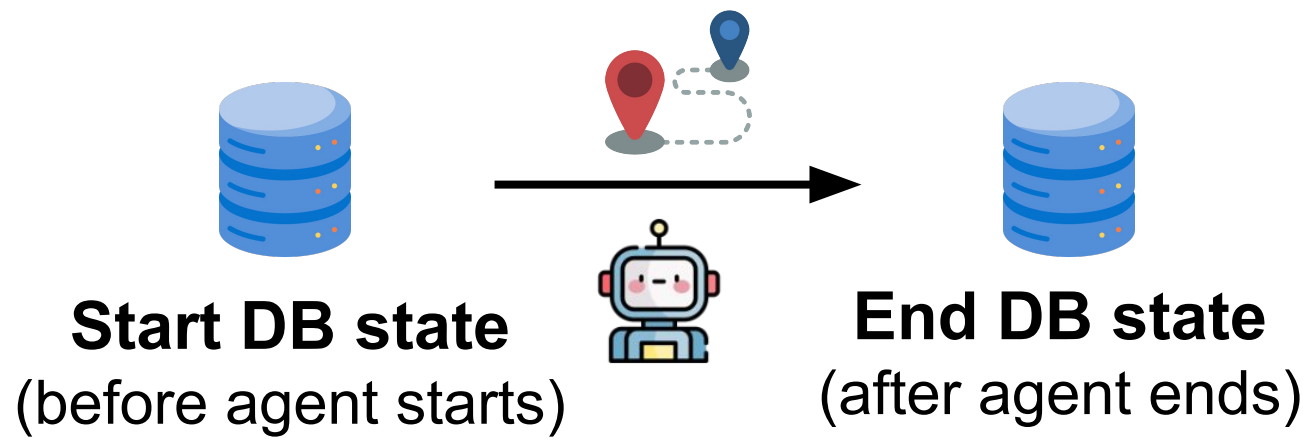


AppWorld uses **State-based** & **Execution-based** approach.

Engine

Benchmark

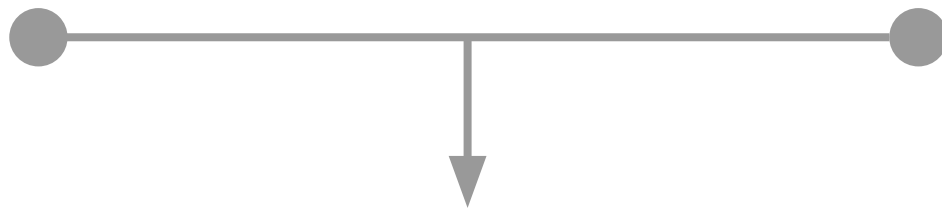
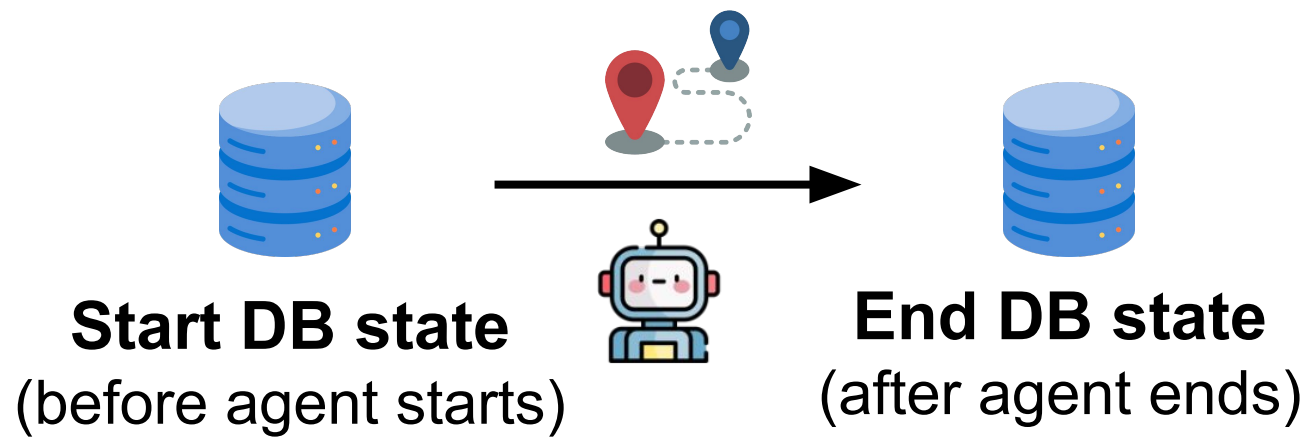
Evaluation



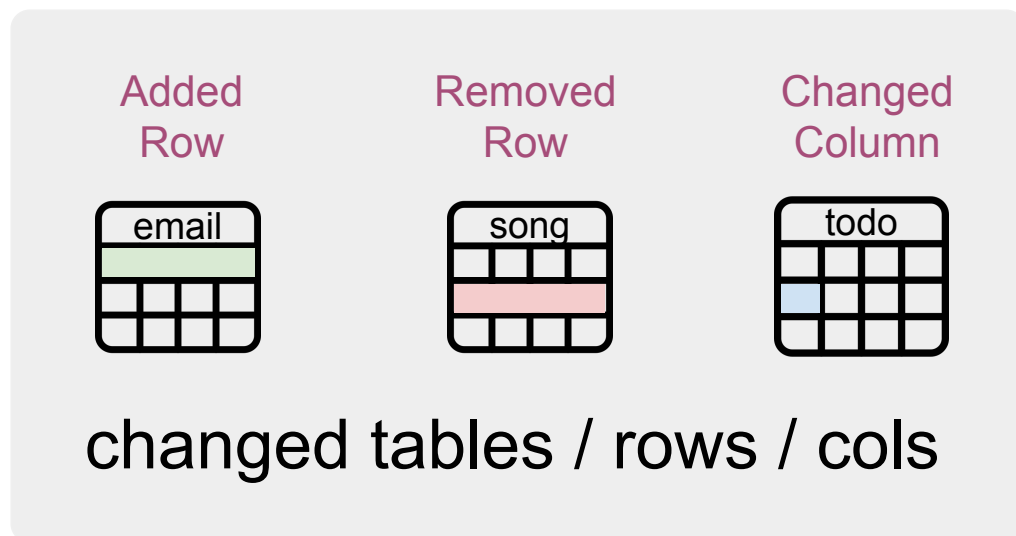
Engine

Benchmark

Evaluation



 Database Difference

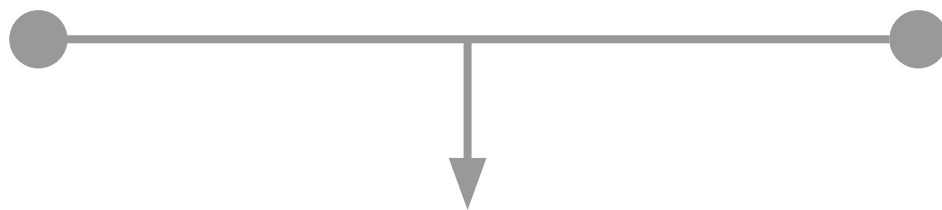
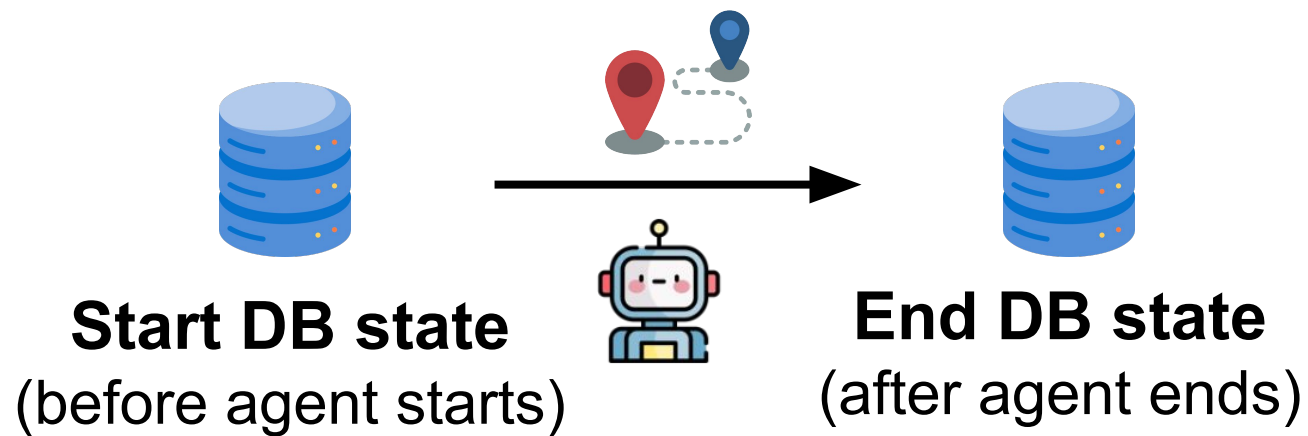



Cumulative DB Changes

Engine

Benchmark

Evaluation




 Database Difference


Added Row Removed Row Changed Column


email	song	todo

changed tables / rows / cols

Cumulative DB Changes


Check Includes?

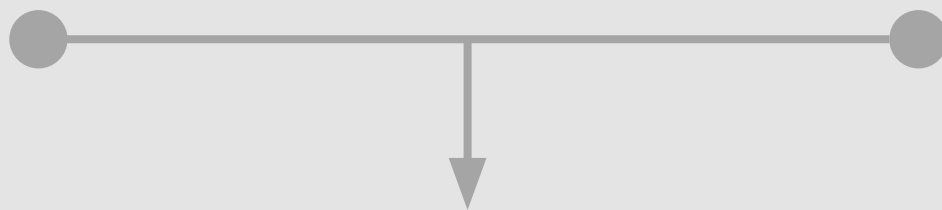
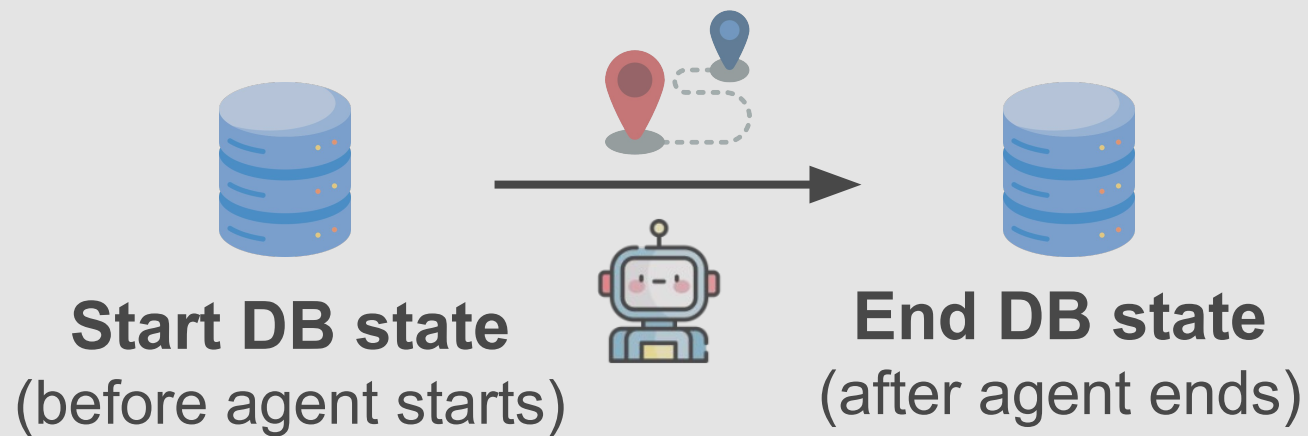
All Expected DB Changes 

No Unexpected DB Changes 

Engine

Benchmark

Evaluation



Programmatic Tests



- ✓ Spotify music player in playing state
- ✓ Player queue duration \geq 45 mins
- ✗ All songs belong to user's playlist.
- ✓ Nothing else except player changed

Database Difference

Added Row

email		

changed tables / rows / cols

Removed Row

song		

Changed Column

todo		

Cumulative DB Changes



Check Includes?

All Expected DB Changes



No Unexpected DB Changes



Evaluation Metrics

1

Task Goal Completion:
All tests in a task pass.

2

Scenario Goal Completion:
All tests for all 3 tasks in a scenario pass.

How do Agents perform on AppWorld?

Open and close LLMs with various few-shot prompting methods.

ReAct

Interleave reasoning thoughts with Code snippet generation.

PlanExec

Generate high-level plan and then execute each of its step via ReAct.

IPFunCall

Iterative calls to Parallel Function Calling

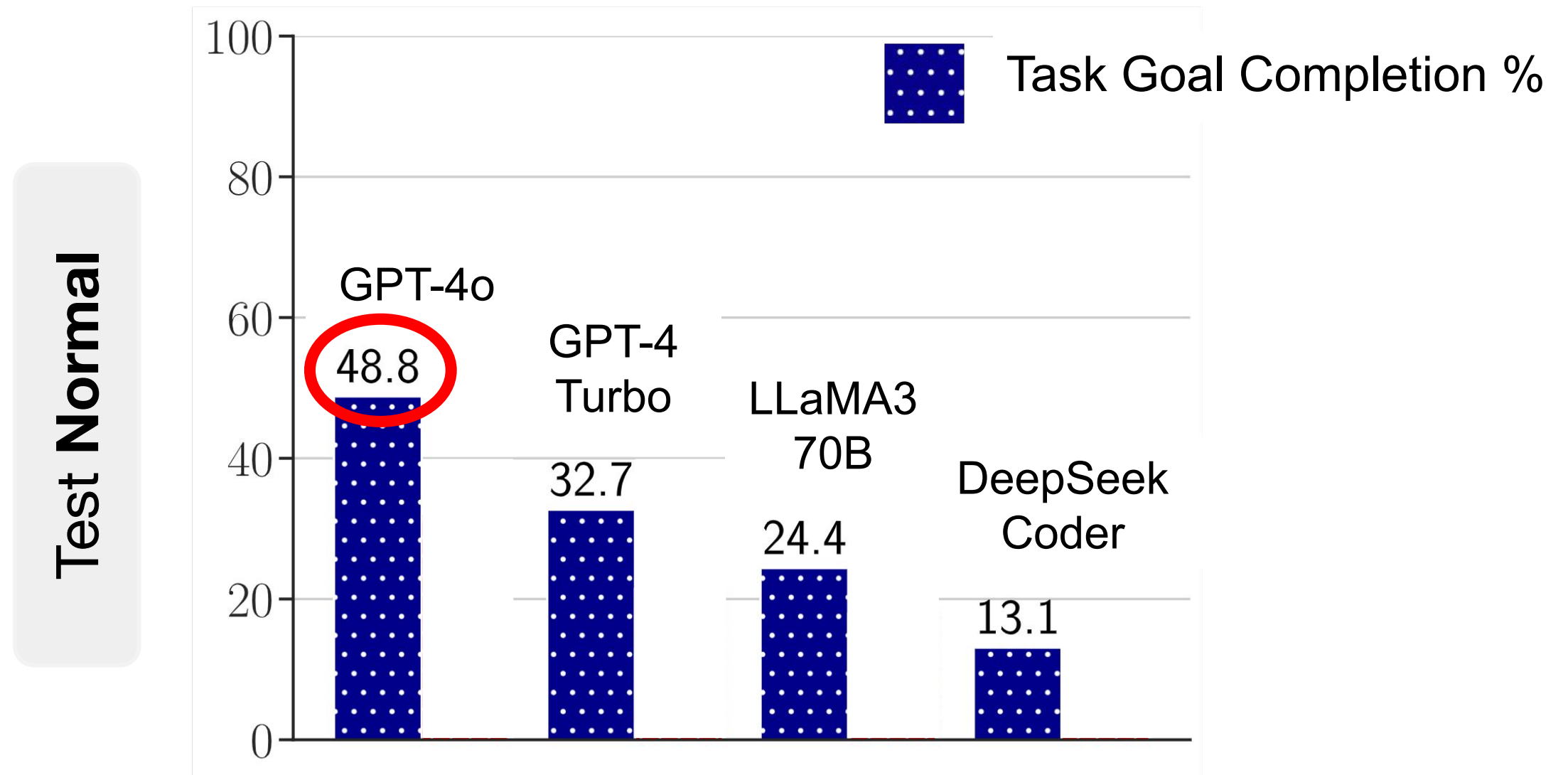
FullCodeRefl

Generate Full Code with multiple rounds of reflexion retrials.

See paper for more agents (e.g., CodeAct, ToolLLM)

How do Agents perform on AppWorld?

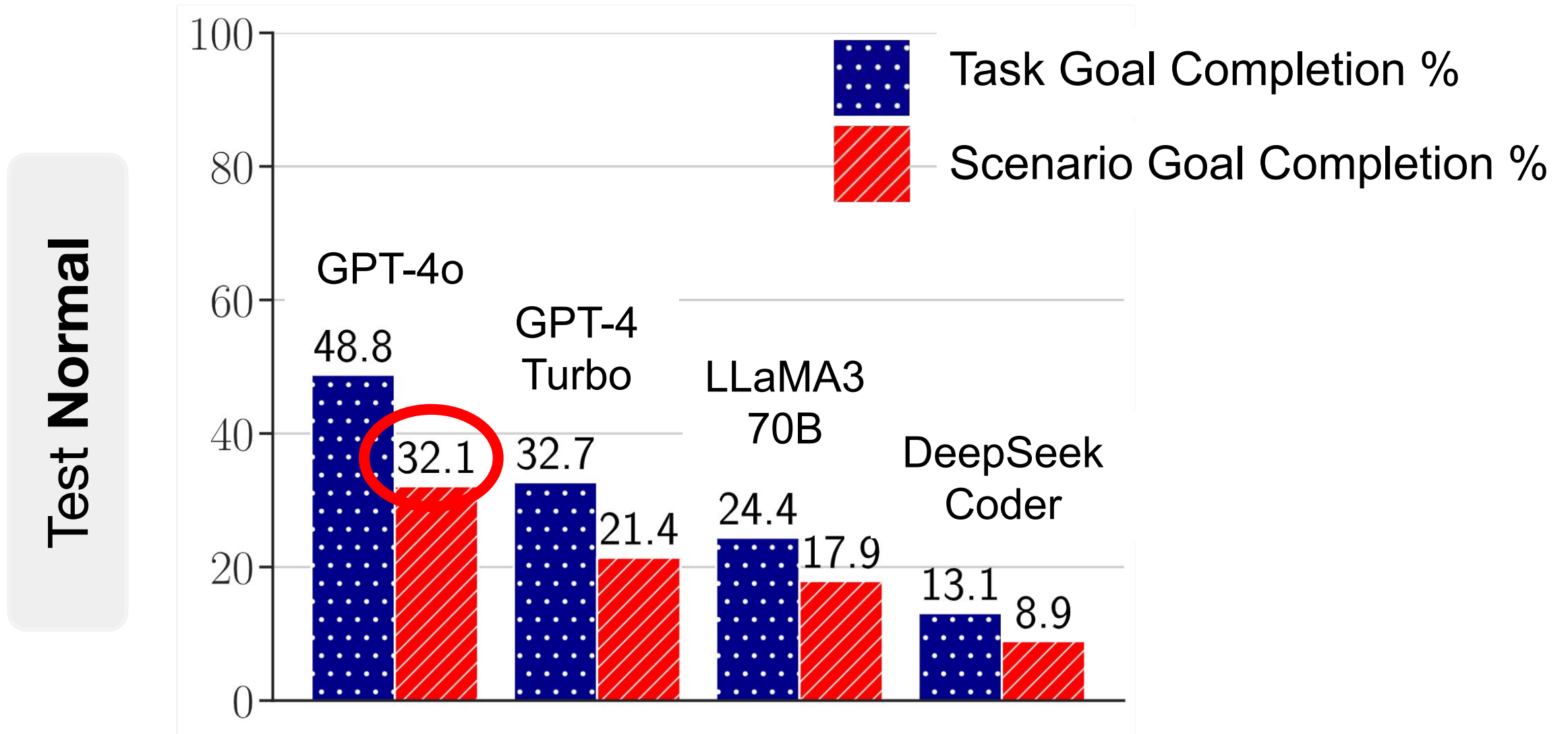
State-of-the-art LLM agents struggle on AppWorld.



For each LM, max score across 4 few-shot methods:
ReAct, PlanExec, FullCodeRefl, IPFunCall

How do Agents perform on AppWorld?

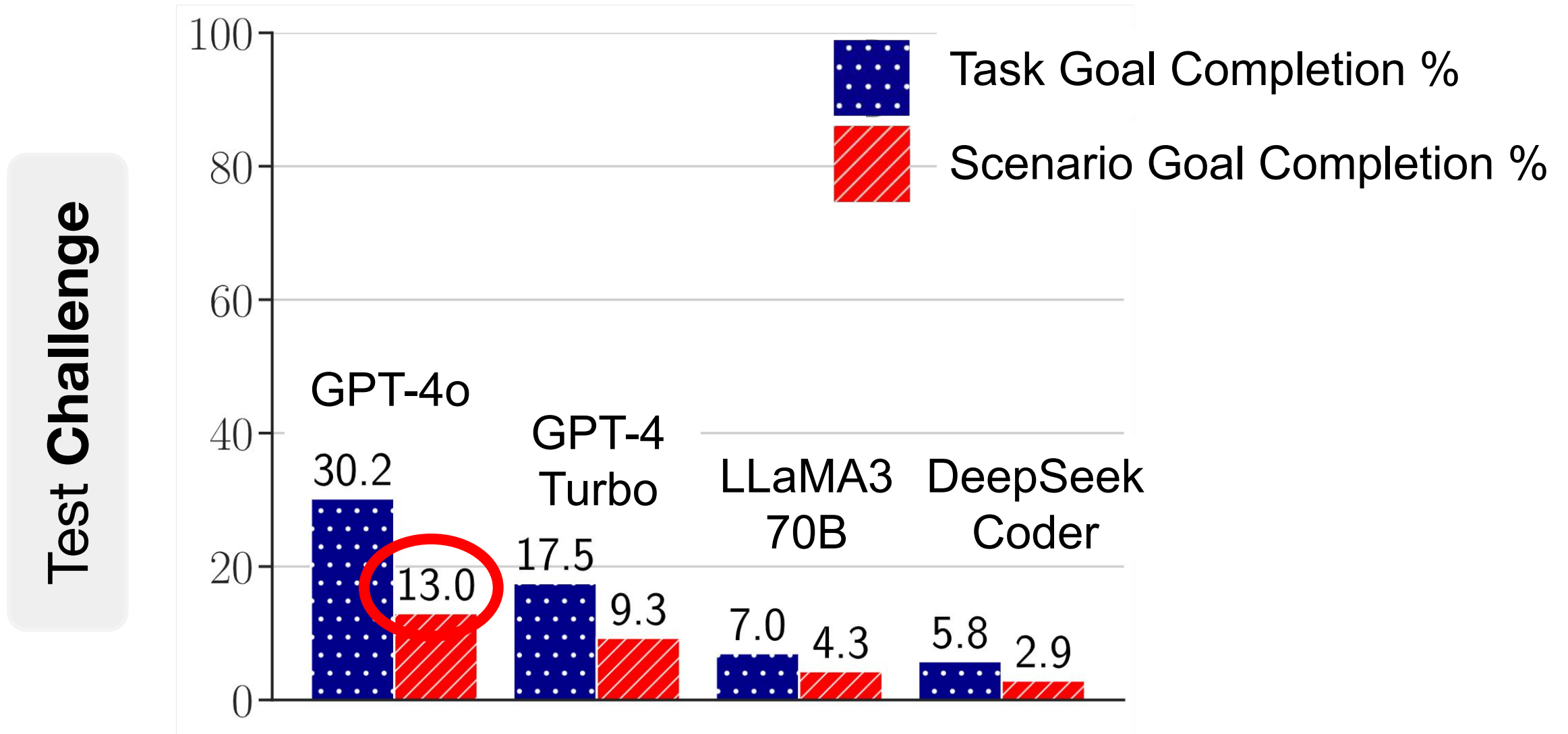
State-of-the-art LLM agents struggle on AppWorld.



For each LM, max score across 4 few-shot methods:
ReAct, PlanExec, FullCodeRefl, IPFunCall

How do Agents perform on AppWorld?

State-of-the-art LLM agents struggle on AppWorld.

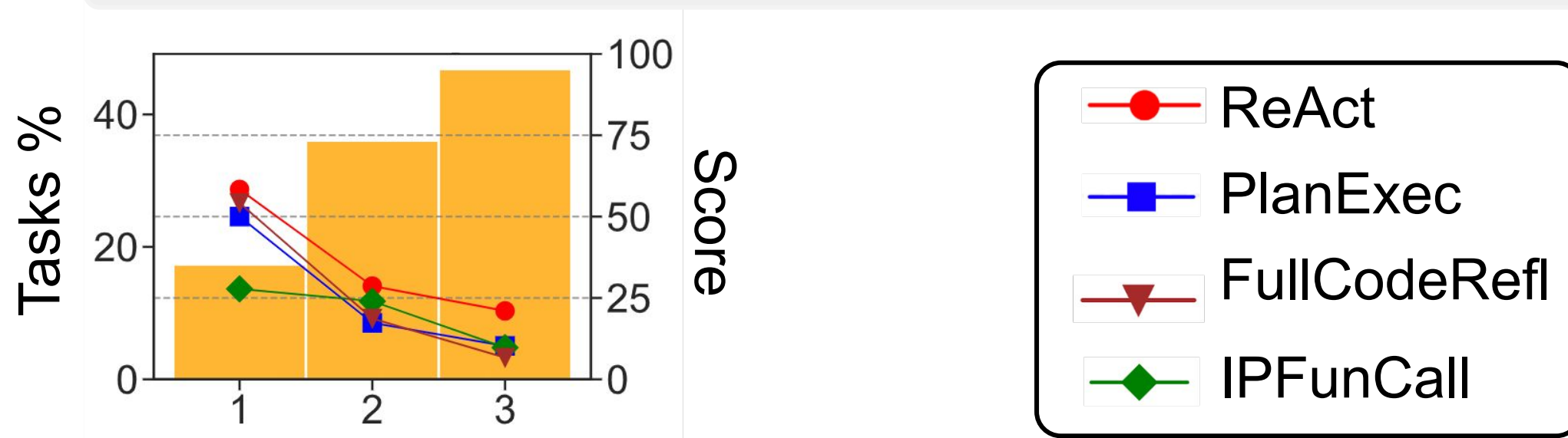


For each LM, max score across 4 few-shot methods:
ReAct, PlanExec, FullCodeRefl, IPFunCall

How do Agents perform on AppWorld?

Agent scores drop drastically as task difficulty increases.

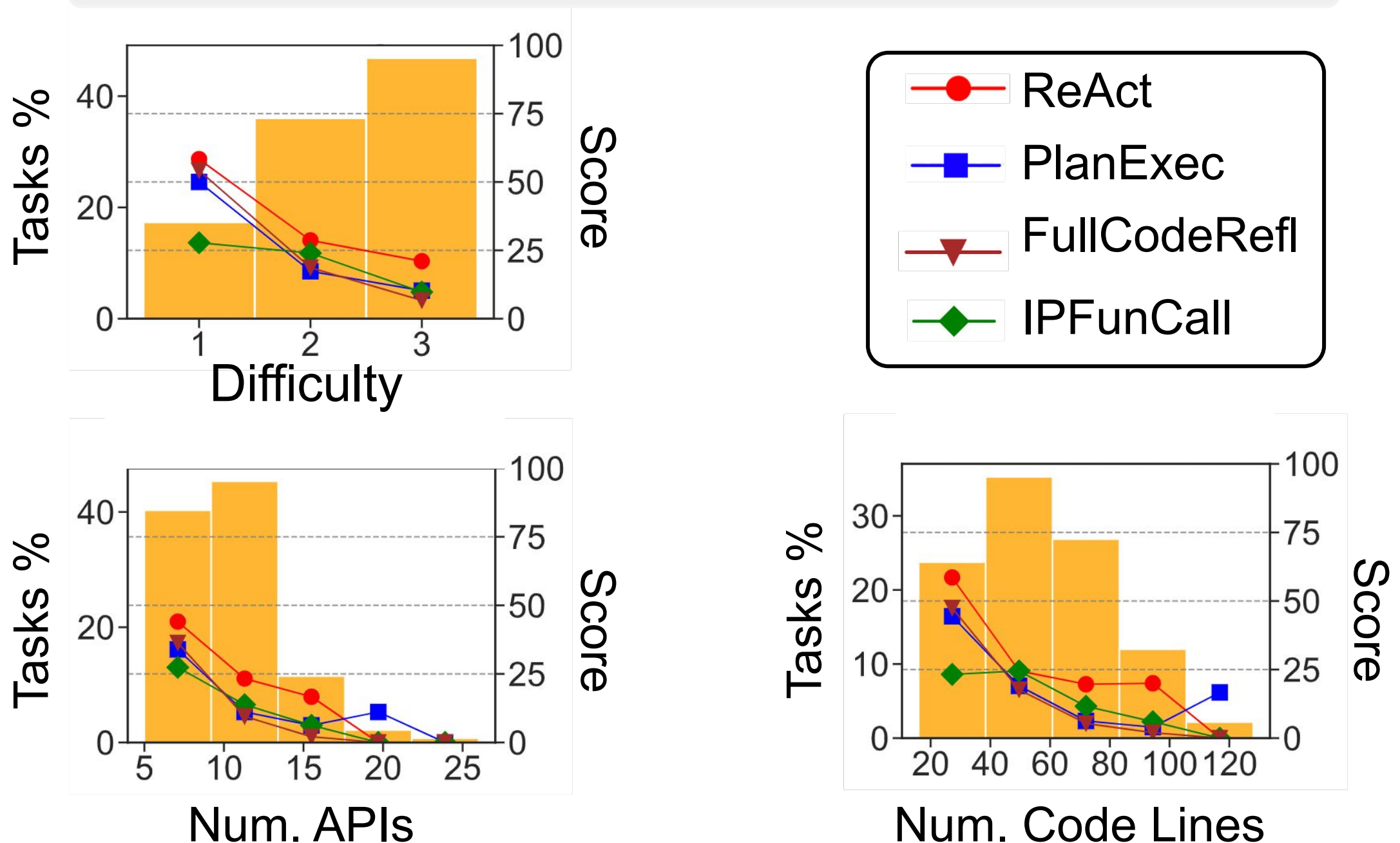
GPT-4o Task Goal Completion % on Test Challenge



How do Agents perform on AppWorld?

Agent scores drop drastically as task difficulty increases.

GPT-4o Task Goal Completion % on Test Challenge



How do Agents perform on AppWorld?

Agent scores drop drastically as task difficulty increases.

Common Agent Errors



Lack of environment interaction



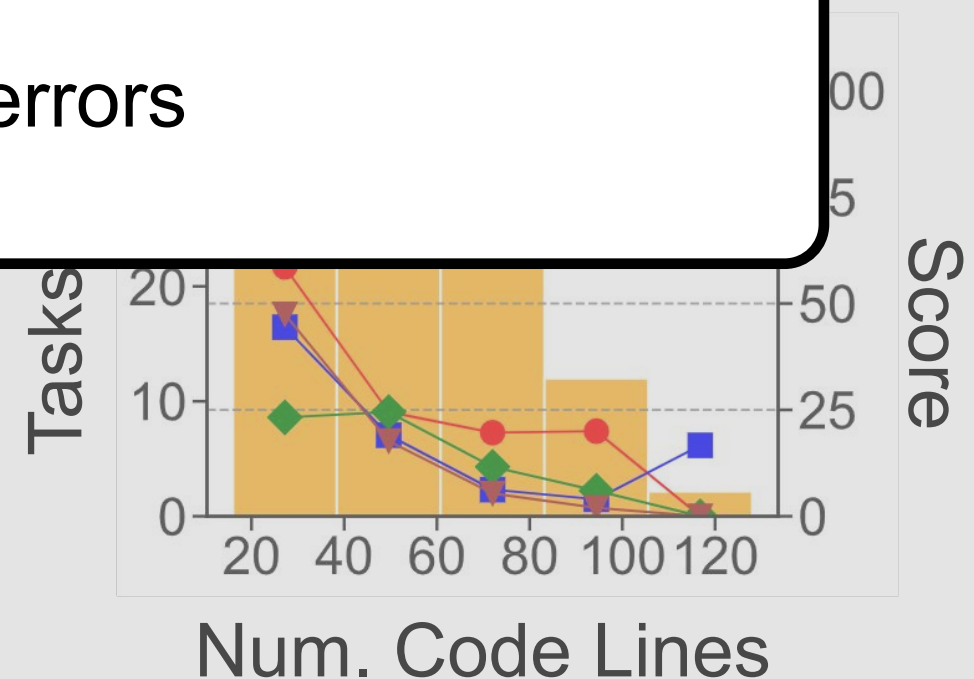
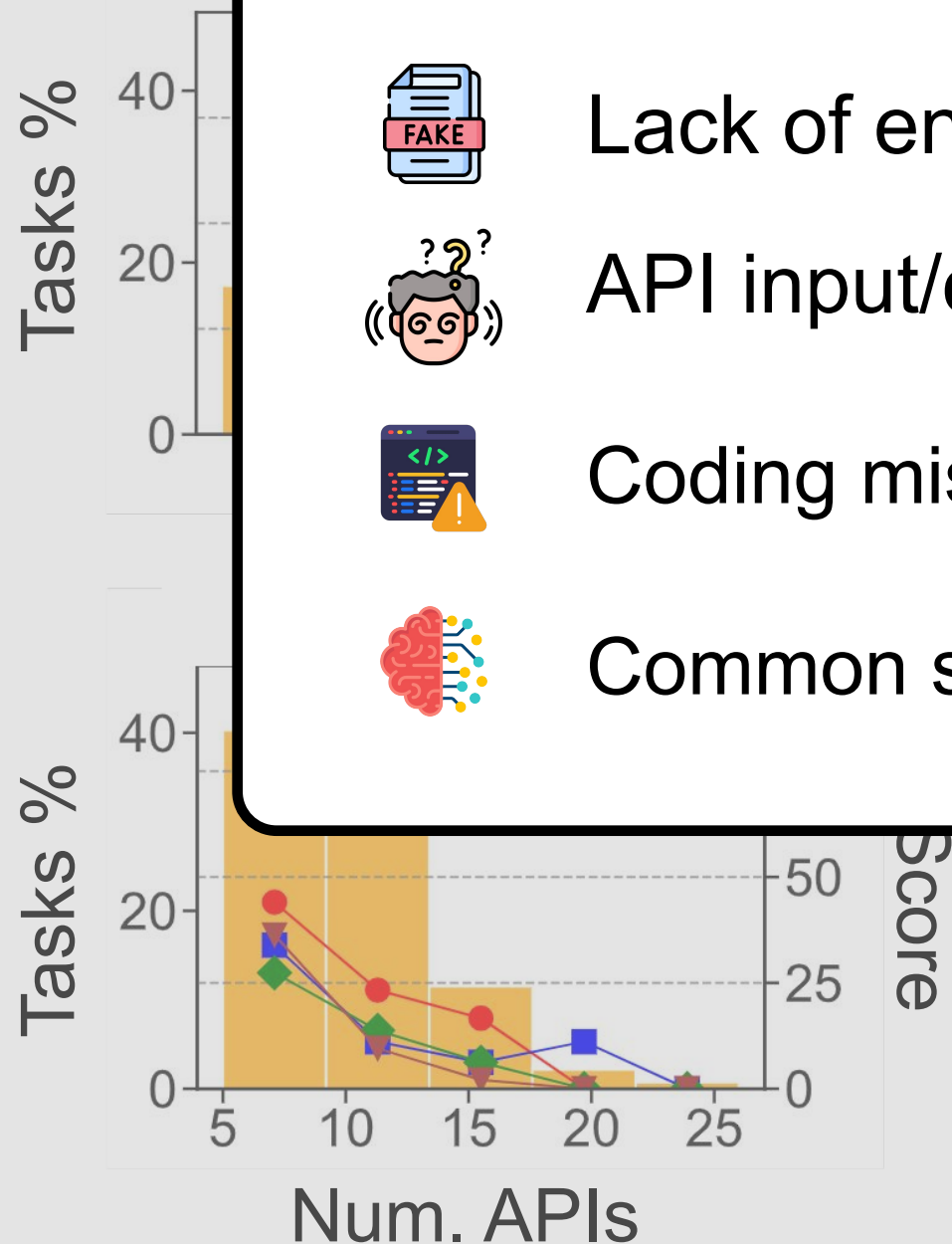
API input/output fields hallucination



Coding mistakes



Common sense errors

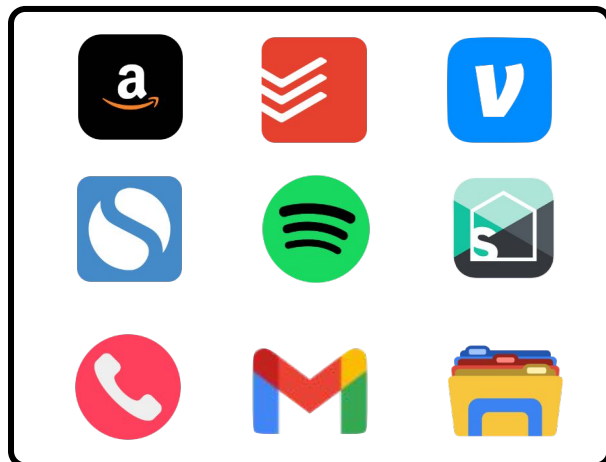


Conclusion

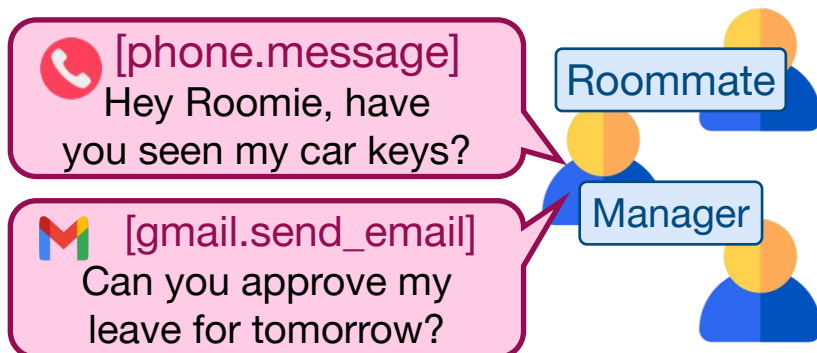


AppWorld

World of Apps & People



- ⇒ 9 app implementations
- ⇒ **High Fidelity**: 457 APIs w/ docs, 100+ tables
- ⇒ **Reliable**: 1700+ unit tests
- ⇒ Simulated **people + activities**



100K+ lines of code

- ⇒ 750 **complex tasks** w/ 5-25 APIs, 1-6 apps per tasks, **rich + interactive coding** (20-130 lines)
- ⇒ Robust programmatic eval

Future Possibilities

Better Agents



Self-exploration

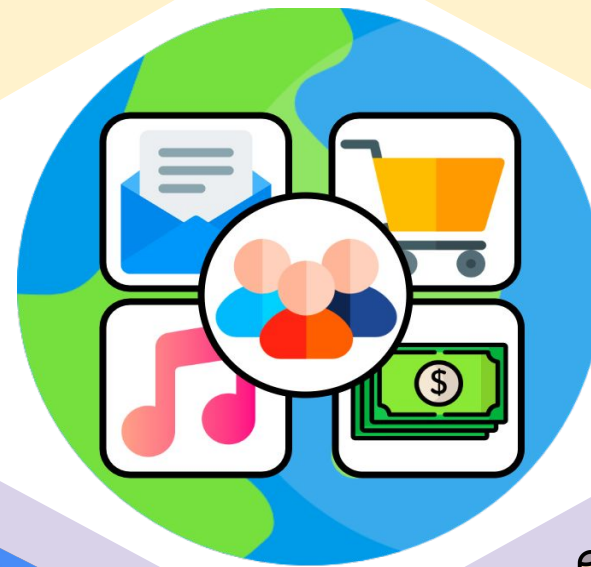


Learning from Feedback

New Agent Benchmarks



UI-based Control
(coming soon!)



Multi-Agent + Human tasks



Study Agents in Environment



Study safety & privacy risks



Study social dynamics of role-playing agents

Task Explorer

API Explorer

Leaderboard

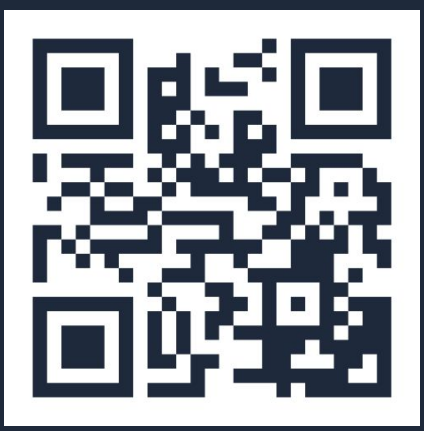
Code

TLDR Video

Tweet

Blog

Paper



<https://appworld.dev>

```

run.py

from appworld import AppWorld, load_task_ids

task_id = load_task_ids("test_normal")[0]
world = AppWorld(task_id=task_id)
agent = YourAgent(world.task)
while not agent.done():
    code = agent.step()
    output = world.execute(code)
    agent.update(output)
world.close()
scores = world.evaluate()

```

Build & Test your Agent

AppWorld is Easy-to-Use!
 Just 'pip install appworld' & start.
 No docker / server necessary,
 Comes with Jupyter-styled shell.

And it is Fast!
 Tasks load in < 0.5s,
 evaluate in < 0.6s,
 & APIs respond in << 30ms.

Poster on
 Tue 4-5:30pm
 in Convention
 Center A1

Few free
 t-shirts!

Extra: How was DB populated

Extra: What is the human score

Extra: What are task variations

Extra: What not use real apps

Extra: Extra More Cmplx Task