MAX PLANCK INSTITUT FÜR INTELLIGENTE SYSTEME, TÜBINGEN

Programmers or Code Charlatans?

LLMs vs. Torch 2.7: Why Your Code Assistant Can't Keep Up 

Diganta Misra

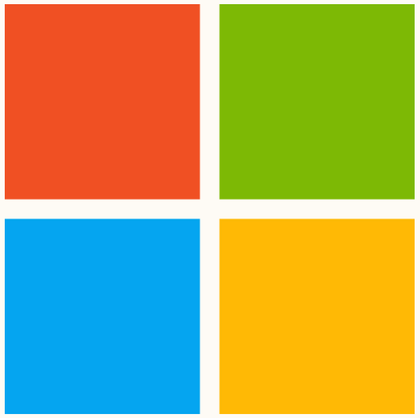
AWS x ELLIS x IMPRS-IS PhD Fellow

SDEs vs. AI

- Code - highly volatile and dynamic
- Library 🔥 → Users ⬆ → Library ⚙ → 🔁
- Imperative: Continual updating of code LLMs knowledge base.
- Imperative: Dynamic benchmarking of code LLMs. (Example - LiveCodeBench[†])

[†] *Jain, Naman, et al. "Livecodebench: Holistic and contamination free evaluation of large language models for code." arXiv preprint arXiv:2403.07974 (2024).*

Onboard the AI hype-train



20-30% internal python code by AI

~Satya Nadella
(CEO, Microsoft)

Salesforce SDE hiring pause, 30% higher productivity with AI

~Marc Benioff
(CEO, Salesforce)

25% new Google code AI generated

~Sundar Pichai
(CEO, Google)

Motivation

Problem Statement

Instruction: Finish implementing the following function to performs a spatial join using the library *geopandas* version *0.9.0*.

```
import geopandas as gpd

def spatial_join(gdf1: gpd.GeoDataFrame,
                 gdf2: gpd.GeoDataFrame)
              -> GeoDataFrame:
    return
```

Attempted Solution

Model: gpt-4o-mini

Solution:

```
gpd.sjoin(gdf1, gdf2, how='inner',
predicate='intersects')
```

Validation Result:

```
TypeError: sjoin() got an unexpected keyword
argument 'predicate'
```

Problem Statement

Instruction: Write a *custom_violinplot* function that visualizes x and y from a Pandas DataFrame; scales the bandwidth to 1.5.

```
import seaborn as sns
from matplotlib.axes import axes

def custom_violinpolot(data: pd.DataFrame)
              -> Axes:
    return
```

Attempted Solution

Model: gpt-4o-mini

Solution:

```
sns.violinplot(x='x', y='y', data=data,
bw=1.5)
```

Validation Result:

```
AssertionError: bw parameter should not be
used. Use bw_method and bw_adjust instead.
```

GitChameleon

Novel, comprehensive Python execution based version conditioning benchmark

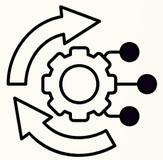
- **328** samples (Torch, GeoPandas, Scipy ..)

Extensive Unit Tests for evaluation

- Hidden and visible tests (**96.5%** coverage)

Misra, Diganta, et al. "GitChameleon: Unmasking the Version-Switching Capabilities of Code Generation Models." DMLR@ICLR (2024).

Collection Protocol



Defining change

Surface level API change

Visible Tests

Manual written assertions

Auto-generated hidden tests

ZenCoder + Self-debug

Defining change

Breaking Change

- Argument or Attribute change
- API call change
- Semantics change
- New feature

Non-candidates

- Nightly versions
- Yanked releases
- Low-level changes
- Numerical stability changes

Structure

Python version: 3.10
Library: scipy
Version: 1.11.1
Release Date: Jun 25, 2023
Example ID: 161
Name of class/func: signal.hilbert
Type of Change: Argument change
Extra dependencies: numpy==1.25.1

Problem:
Complete the function computehilbert_transform.
We are using numpy 1.25.1.

Starter Coder:

```
# library: scipy
# version: 1.11.1
# extra_dependencies: ['numpy==1.25.1']
import numpy as np
from scipy.signal import hilbert

def compute_hilbert_transform(a, b, dtype=np.float64):
  # should return the Hilbert transform of the
  # a and b arrays stacked vertically,
  # with safe casting and the specified dtype.
  # raise TypeError if needed
```

Expected Solution:

```
stacked = np.vstack((a, b), dtype=dtype,
casting="safe")
return hilbert(stacked)
```

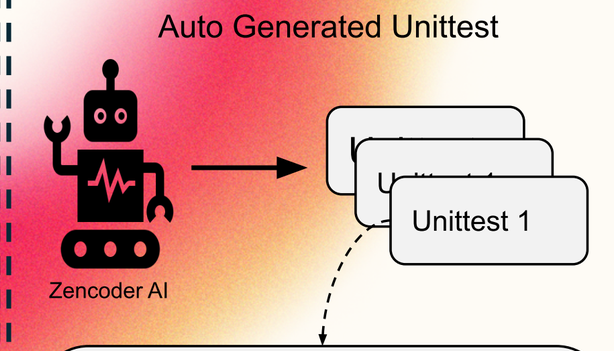
Manual Test:

```
import numpy as np
from sample_161 import compute_hilbert_transform
a = np.array([1.0, 2.0, 3.0], dtype=np.float32)
b = np.array([4.0, 5.0, 6.0], dtype=np.float64)

computed = compute_hilbert_transform(a, b,
dtype=np.float32)
expected = hilbert(np.vstack([a.astype(np.float64),
b.astype(np.float64)])).astype(d
type=np.complex64)

assert np.allclose(computed, expected) &
(computed.dtype == np.complex64)
```

Manually Crafted



Auto Generated Test:

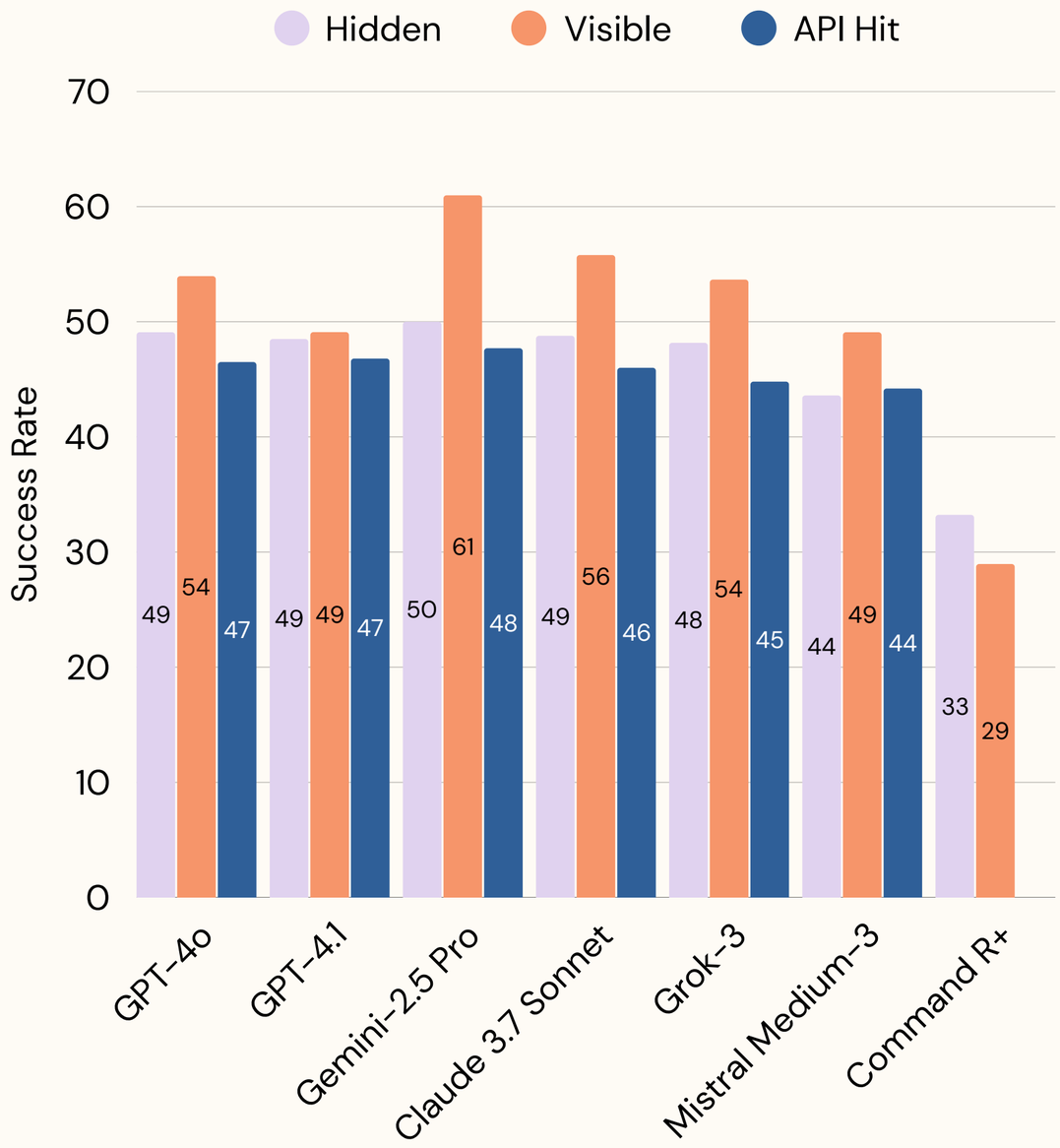
```
def test_basic_functionality(self)
def test_different_dtypes(self)
def test_safe_casting(self)
def test_type_error(self)
def test_different_shapes(self)
def test_multidimensional_arrays(self)
```

Auto Generation

Results

LLMs are version sensitive?

Back to Overview

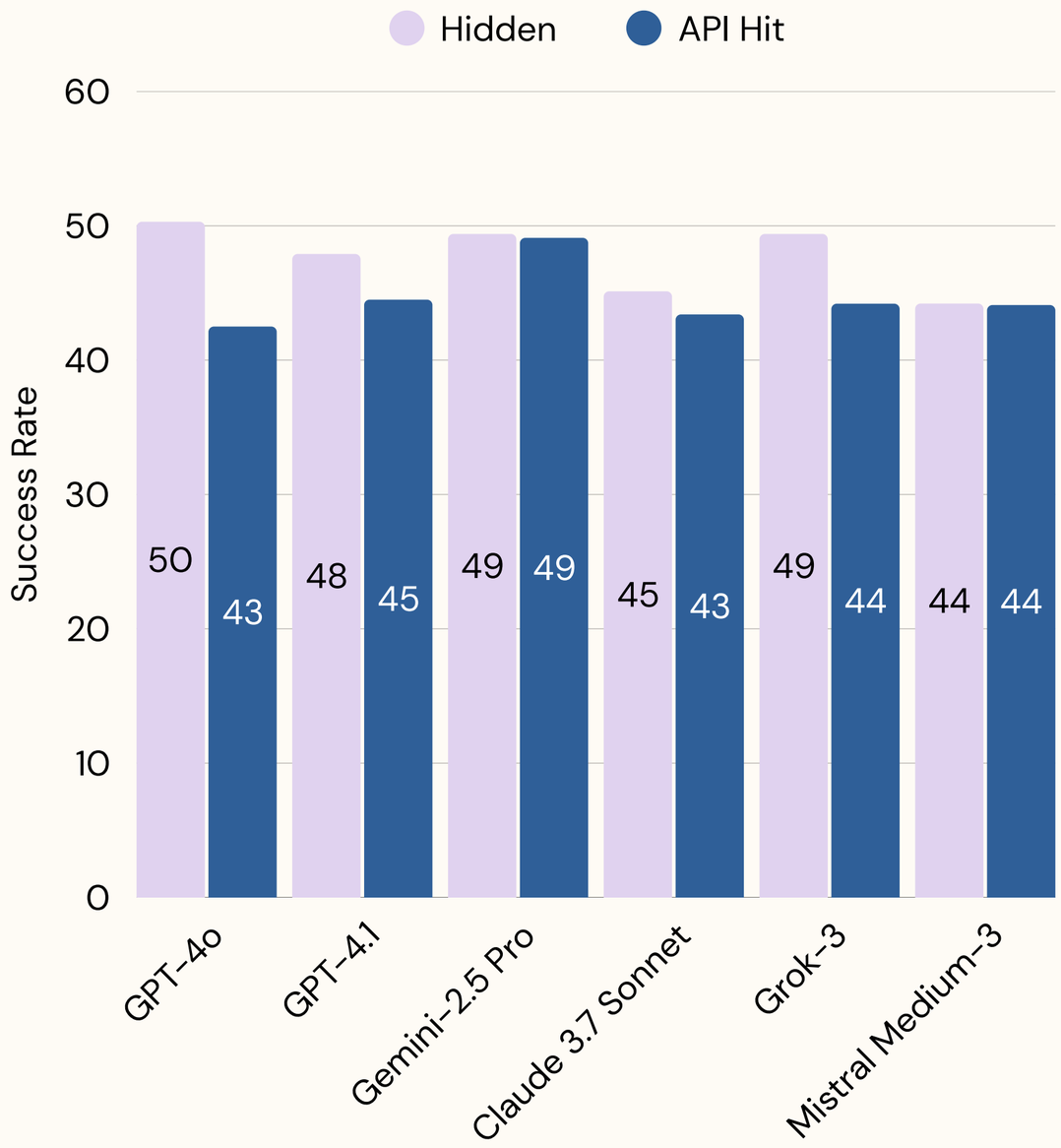


Enterprise

Greedy Sampling

- Best performing model: Gemini-2.5 Pro
- Worst performing model: Mistral Medium-3 (Actual: Command R+)

Back to Overview

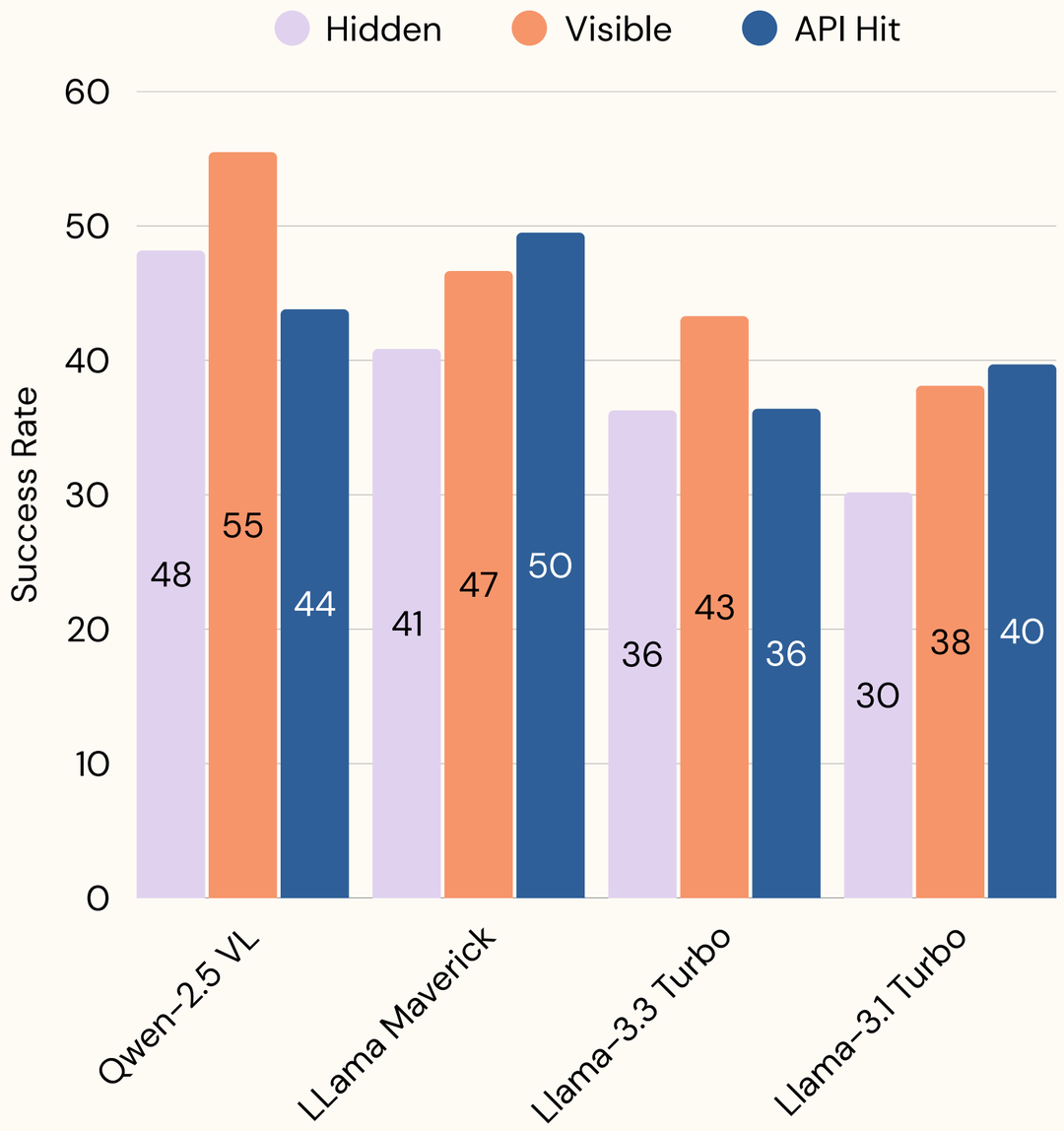


Enterprise

Zero-shot CoT

- Best performing model: GPT-4o (Actual: o3-mini)
- Worst performing model: Mistral-Medium 3 (Actual: GPT-4.1 Nano)

Back to Overview

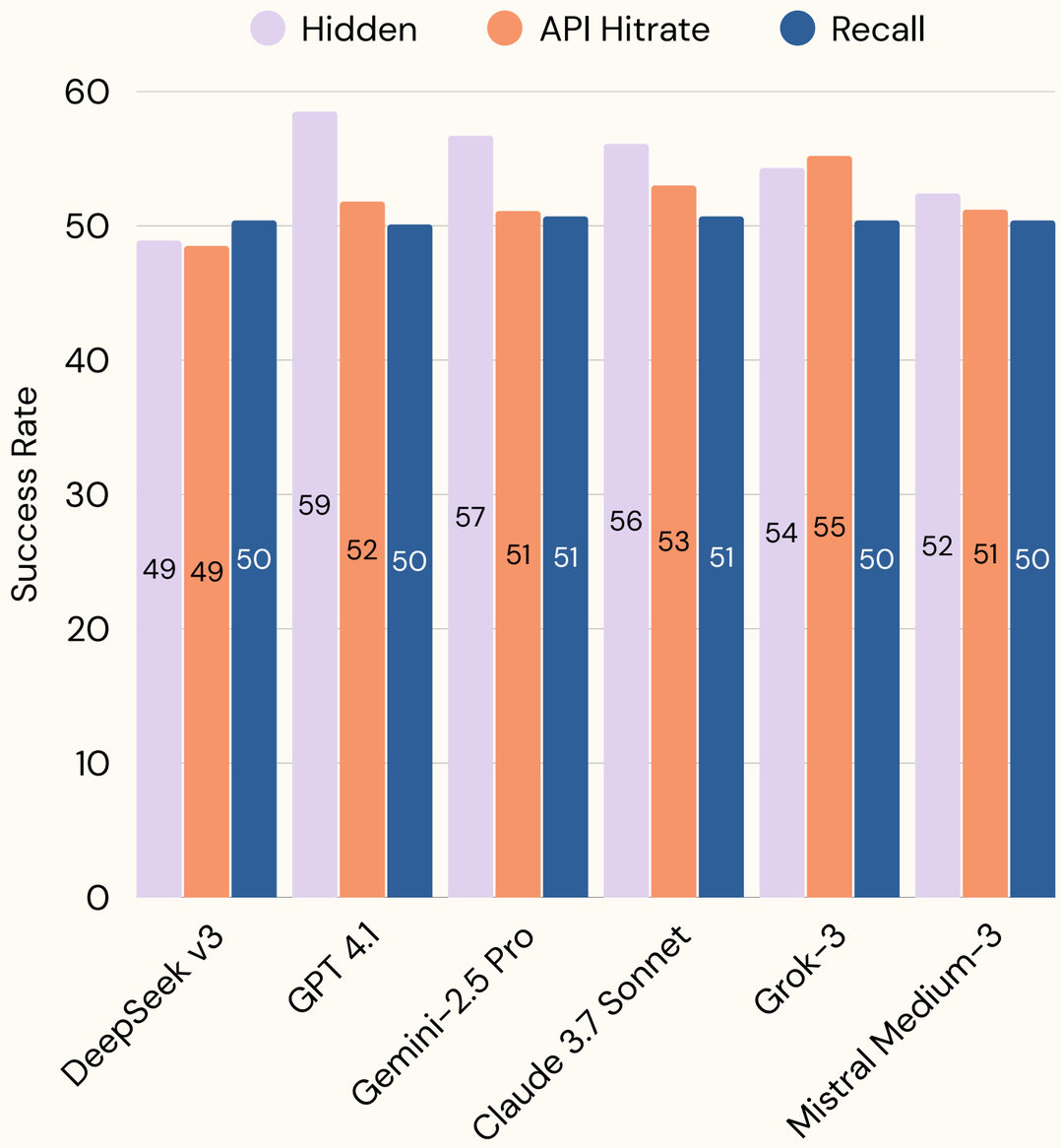


OSS

Greedy Sampling

- Best performing model: Qwen-2.5 VL
- Worst performing model: Llama-3.1 Turbo

Back to Overview

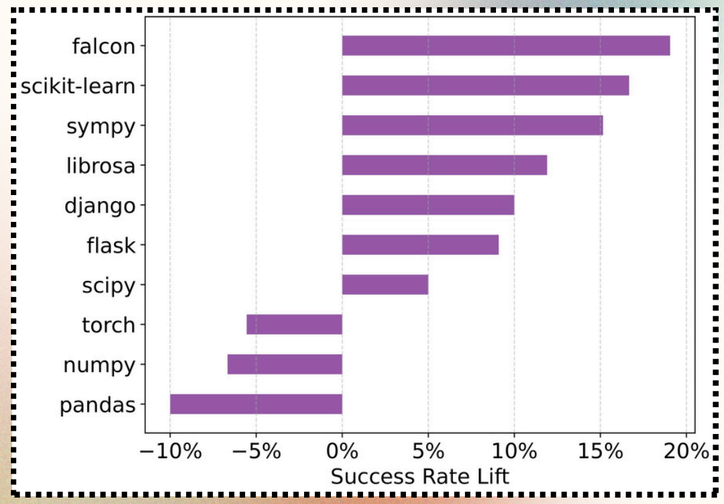


RAG

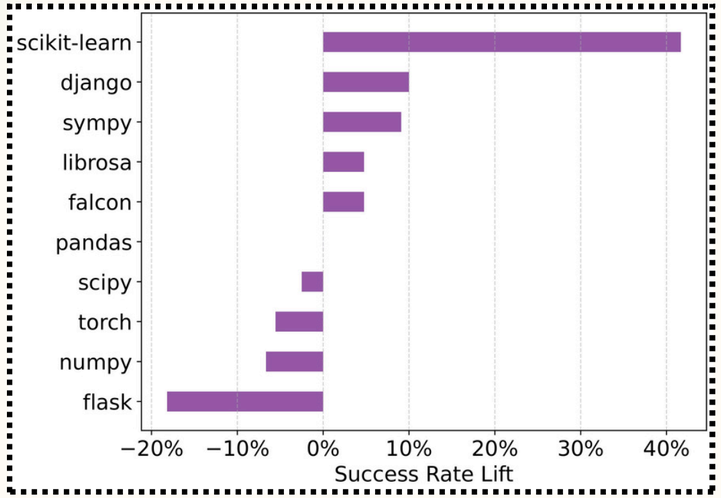
Top-3 retrieved documents

- Best performing model: GPT-4.1
- Worst performing model: DeepSeek-v3 (Actual: Command-R 7B)

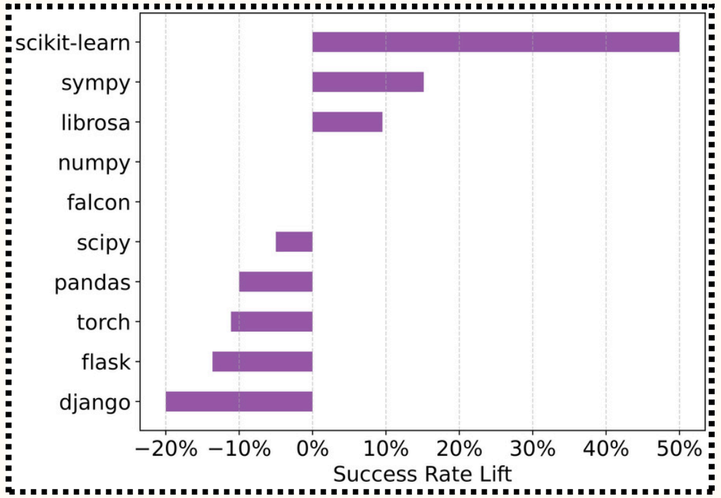
Larger Model = Stronger RAG benefits



GPT-4.1

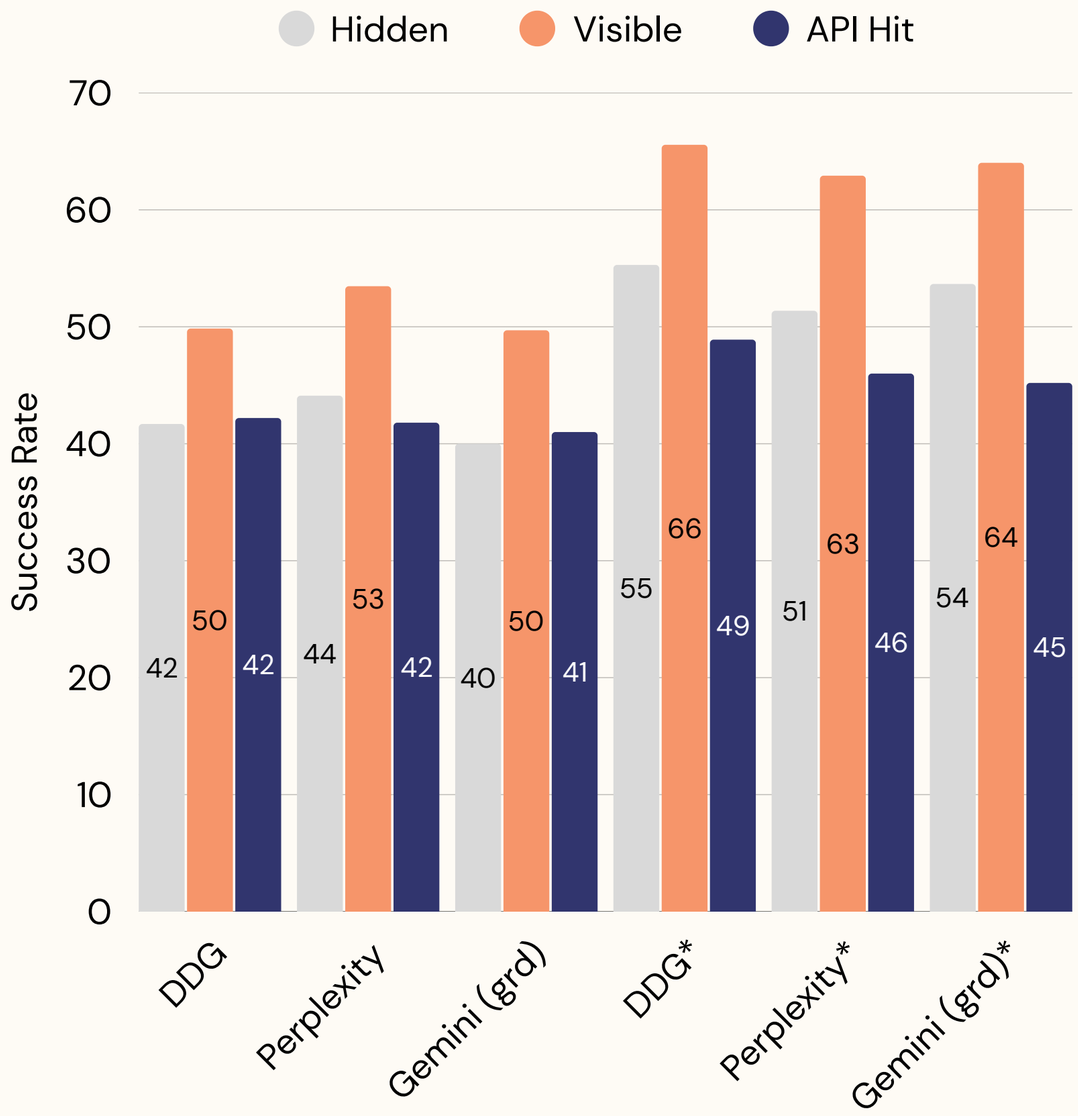


GPT-4.1 Mini



GPT-4.1 Nano

Back to Overview

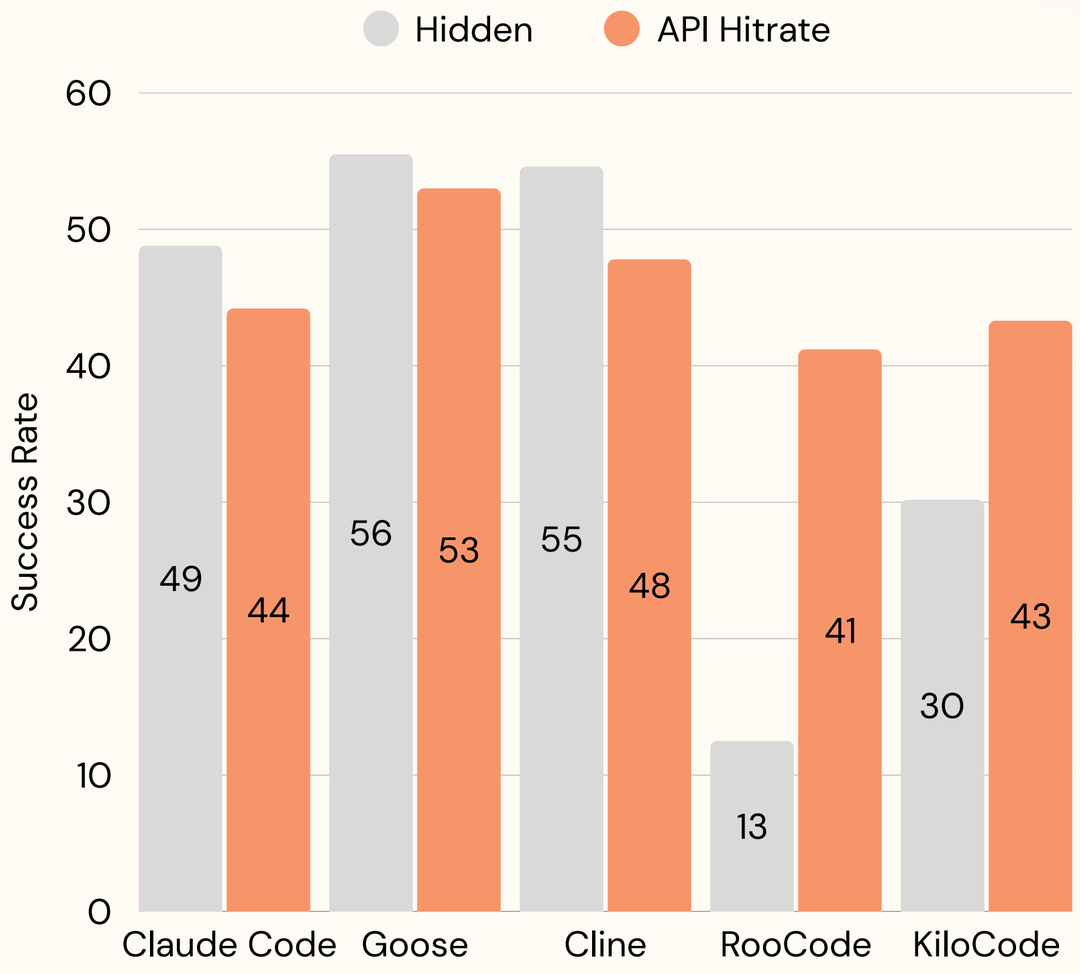


Agents

Base Model: Claude 3.5 Sonnet

- Best performing config: DDG*
- Worst performing model: Gemini + Grounding
- * represents SandBox tool

Back to Overview

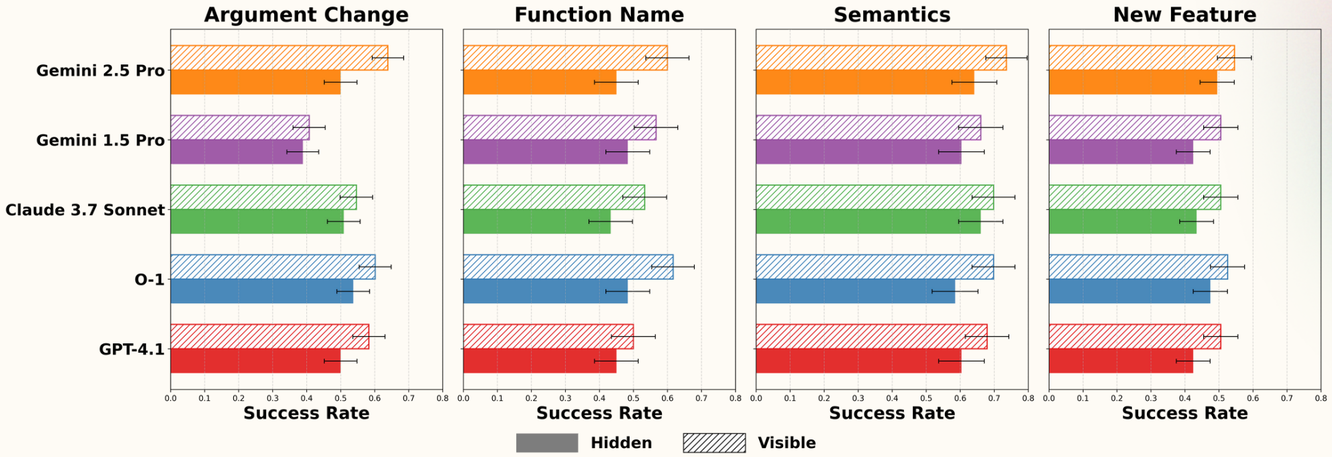


Code Asst.

CLI + IDE Assistants

- Claude Code, KiloCode = Claude 3.7 Sonnet
- RooCode = Claude 3.5 Sonnet
- Goose, Cline = GPT 4.1
- IDE Assistant generation with chat based auto tab completion.
- All VSCode IDE integrations.
- RooCode and Kilocode not exposed to the NL problem statement

Analysis

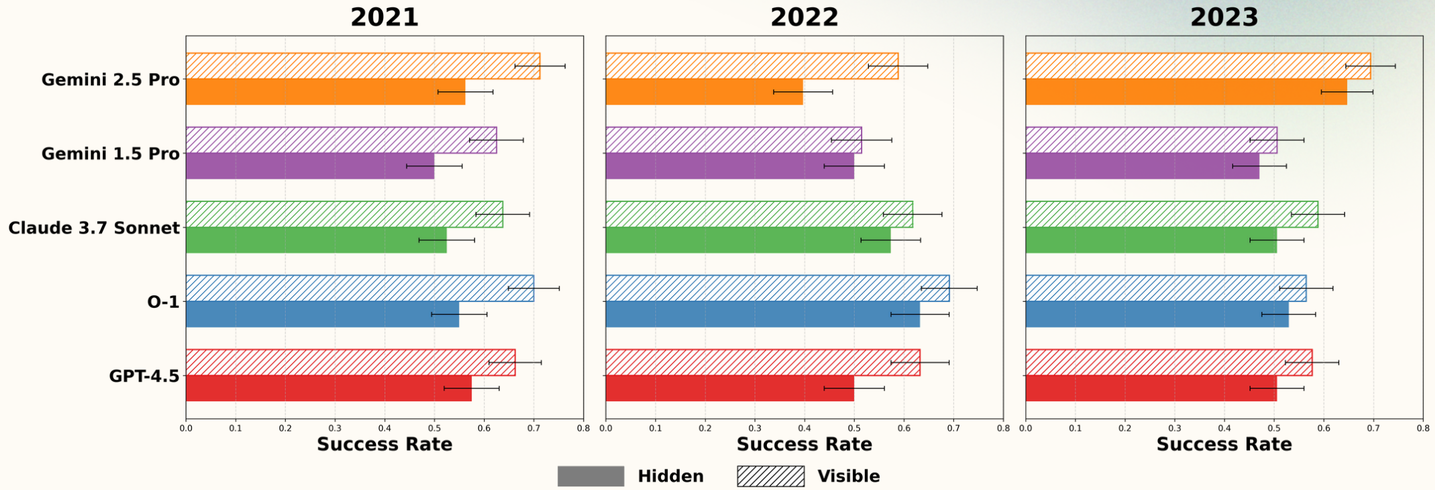


Performance over type of change

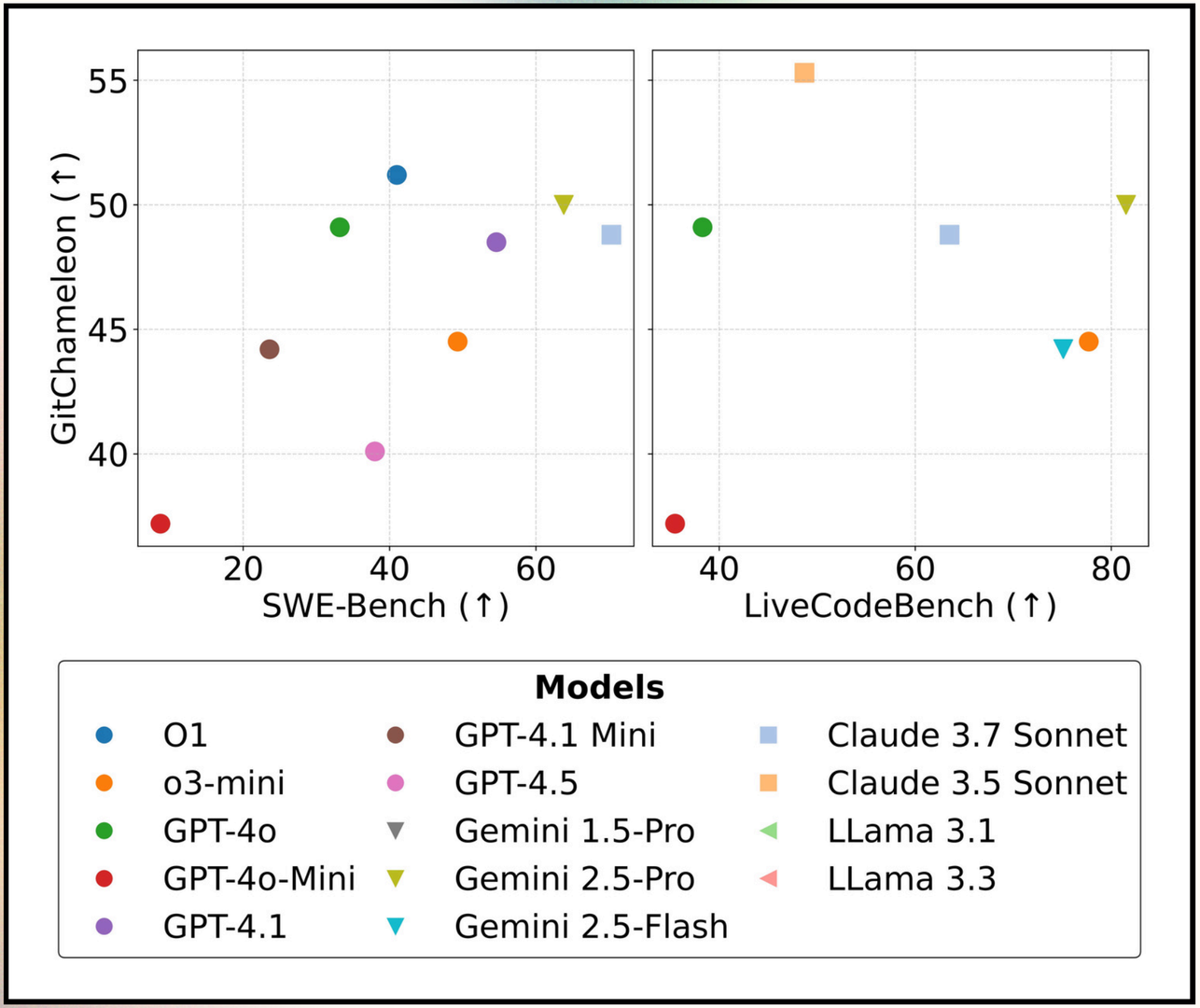
Feature addition in libraries is the biggest obstacle.

Performance over the years

2021 remains as the year to beat likely because Stackv2 has a lot of code from 2021.



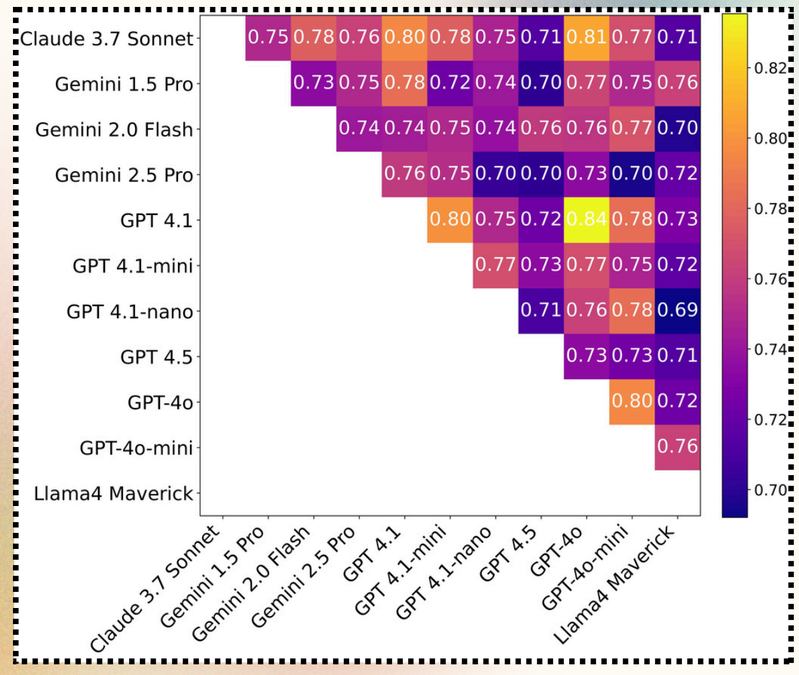
Bootstrapping benchmarks



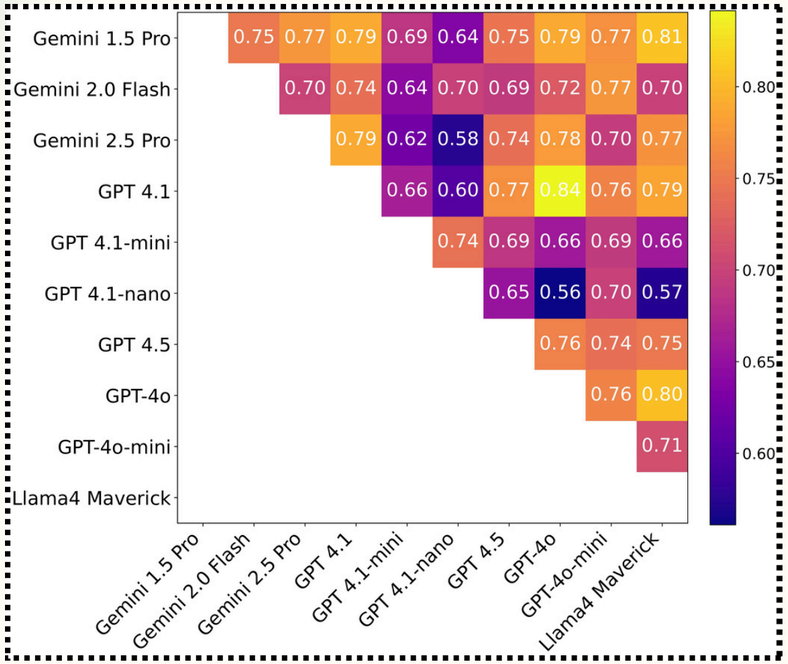
Is GitChameleon perf. correlated with other benchmarks?

Not necessarily, GitChameleon exhibits a moderate correlation with SWE-Bench, with ρ of 0.550 and r of 0.675; and a weak correlation with LiveCodeBench, with ρ of 0.214 and r of 0.130.

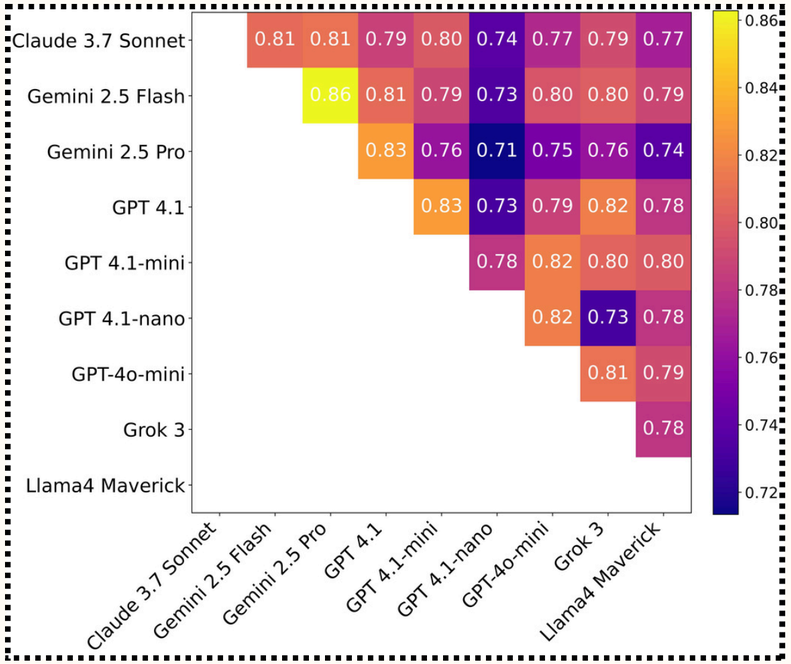
Bootstrapping by ensembling?



Greedy Decoding



Zero-shot CoT



RAG

Summary and Conclusion

Version Sensitivity is desideratum

Having AI models adapt to dynamic changes to libraries

Current AI models/ agents and assts. are not version sensitive

Fail to adhere to specific target versions.

GitChameleon

Our benchmark allows principled eval of such capabilities of code LLMs.

Q&A Session

Thank you for listening!