Research Paper

Understanding the impact of a Semantic Layer in Agentic API Orchestration accuracy

orbital

Introduction

Introduction

How good are LLM’s at **building a plan** requiring **orchestrating multiple APIs** together?

We recently completed a large scale study into the accuracy of leading LLM’s at planning orchestration under real-world conditions.

This paper presents the research, the methodology, and results.

Research Goal

We wanted to answer three key questions:

1 How well do LLM’s perform at API orchestration when facing real-world complexity (hundreds of endpoints)?

2 Does adding semantic metadata improve their accuracy?

3 Does the adoption of a declarative orchestration language make a difference?

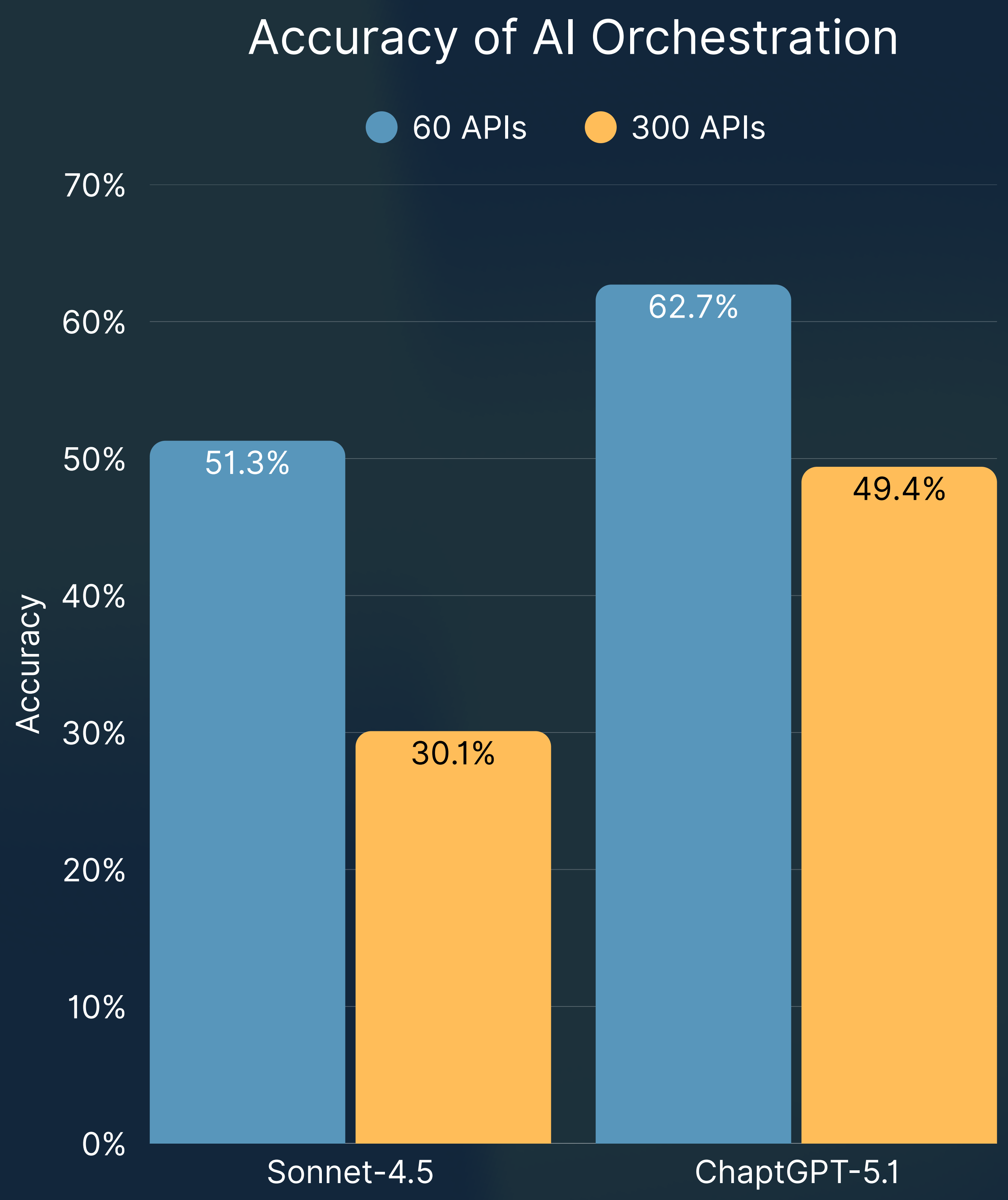
Executive Summary

4 key takeaways from our research

1

Planning accuracy falls to unusable levels between 60 to 300 endpoints

When having to select from 300 endpoints, flagship LLM’s failed our tests in ~70% of test runs

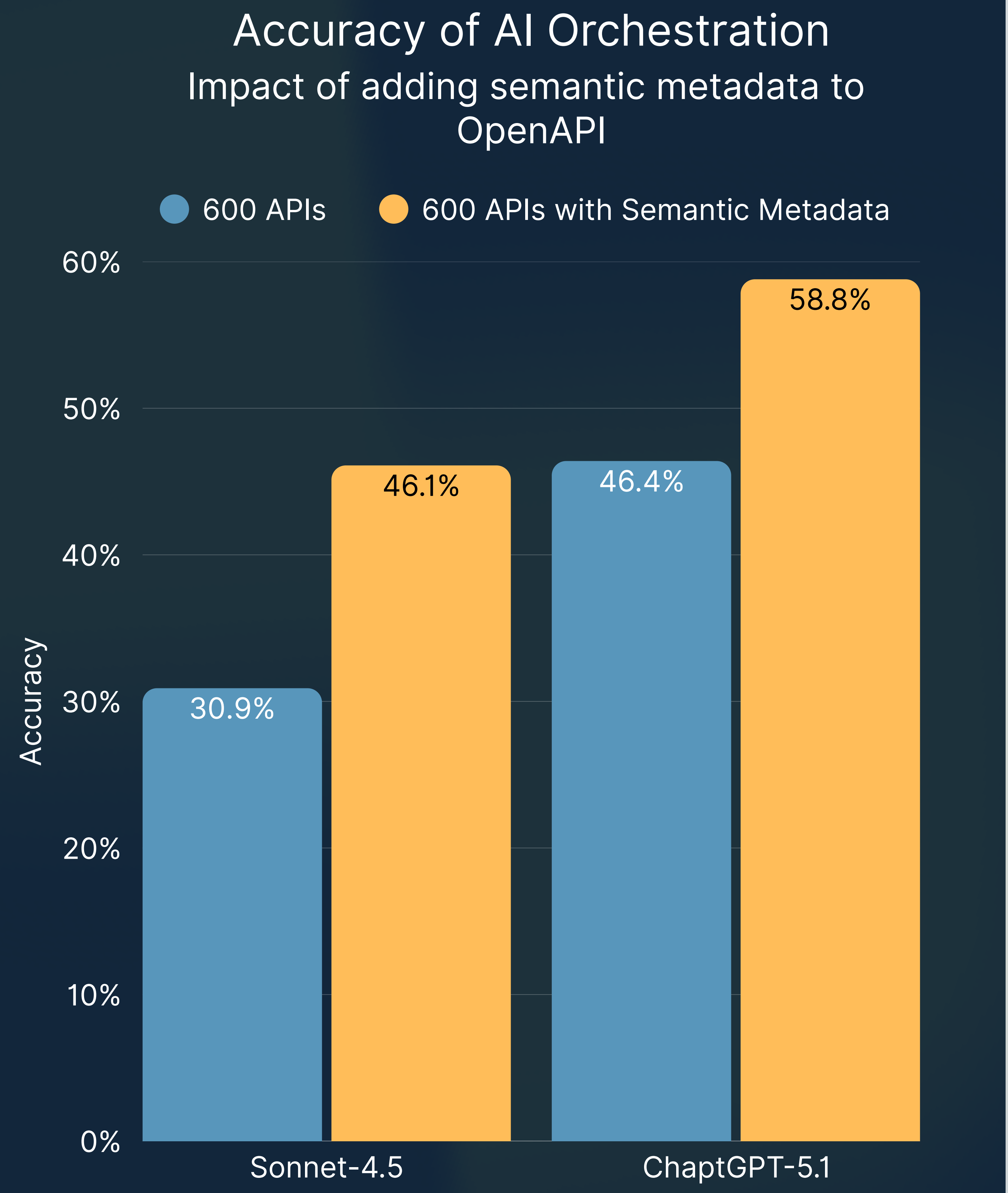


orbital

2

Adding even minimal semantic metadata improves planning accuracy

A lightweight step teams can take today with immediate, measurable impact on LLM planning accuracy



orbital

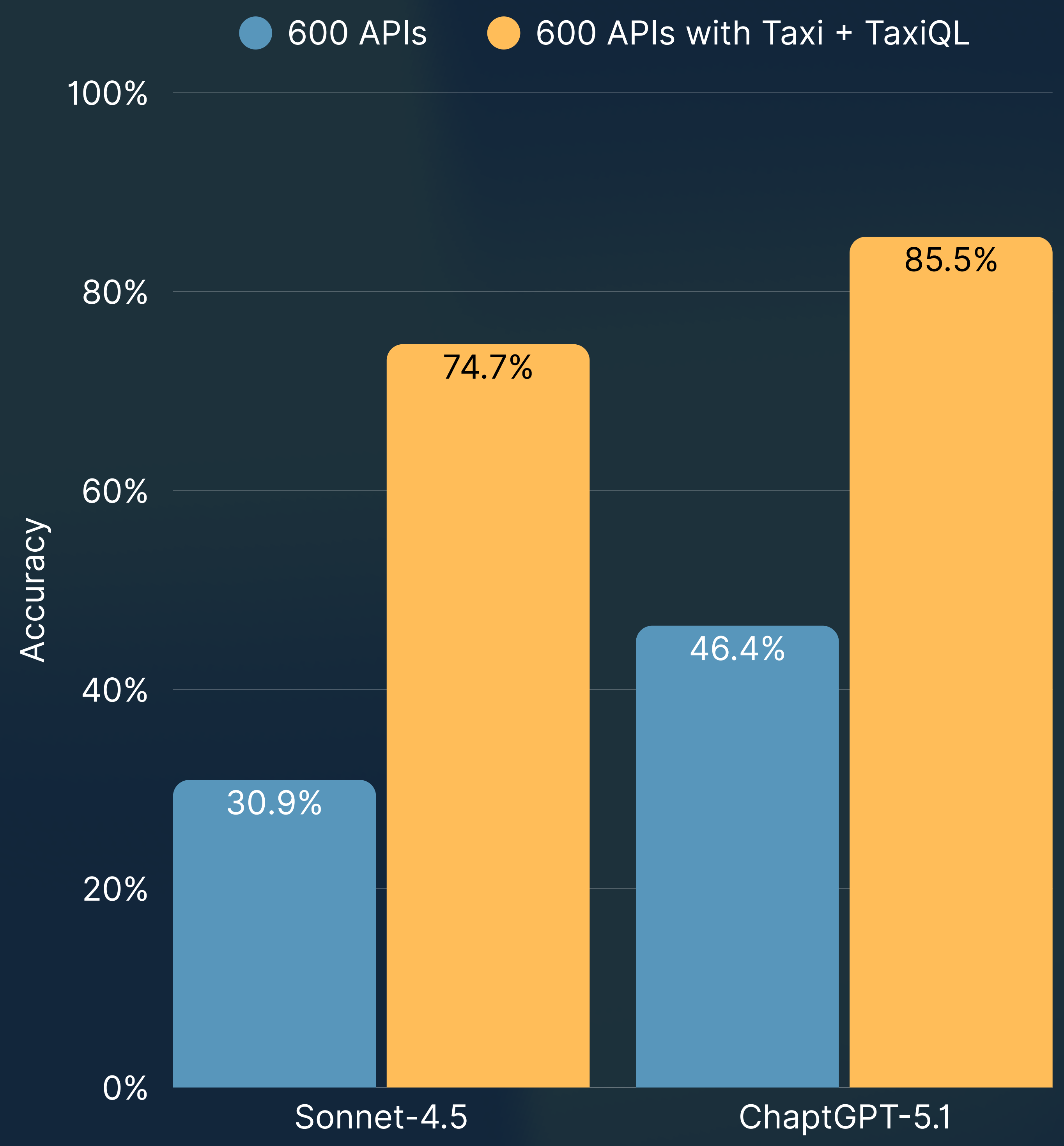
3

Adopting a semantic layer improves LLM planning accuracy by 73-142%

AI agents using declarative query languages (like TaxiQL) to express intent dramatically outperformed LLMs generating orchestration code directly

Accuracy of AI Orchestration

Raw agentic planning vs using Taxi+TaxiQL



4

Using Taxi to describe data sources reduces token usage by up to 80%

Taxi is an alternative schema language.

Adoption results in reduced token usage (driving down cost), while also improving LLM performance via higher-density context.

The Task

Building a plan to call the right APIs in the right order

Methodology - Task

LLMs were provided with a collection of OpenAPI specs (tested at multiple volumes) and a development task that had to be implemented.

The solution required orchestrating the right 5 APIs in order.

LLMs needed to tackle real-world challenges, such as:

- Identify the correct endpoints to call
- Recognize and resolve different ID schemes across APIs
- Select the correct data attributes from complex response objects

Methodology - The Task

LLM's were provided an email outlining requirements from a ficticious bank. This scenario was designed to mirror real-world trading workflows at tier-one banks.

> **From**: Sarah Angle-brackets, Senior Equity Trader
> **To**: Development Team
> **Subject**: Need trade impact checker before I submit orders
>
> Hi team,
> I need a tool that helps me understand what's going to happen before I execute large trades.
>
> Right now I submit orders blindly and sometimes run into problems after the fact.
> Here's what I need to know before I click "submit":
>
> - What's my current position in this stock and what will it be after the trade?
> - Will this trade push me over any of my risk limits? I don't want compliance breathing down my neck
> - What's the market telling me - am I going to get a good fill or will I move the market?
> - Are there any corporate actions coming up I should know about? (Last month I bought right before a dividend ex-date and didn't realize)
> - What's this going to do to my VaR? Risk management asks me about this all the time
> - The input would just be what I'm planning to trade - the stock (I usually work with tickers or sometimes I have the ISIN from a client), how much I want to buy or sell, and on which book.
> - The output should tell me if it's safe to proceed or if I need to get approval first. I want to see everything in one place so I don't have to check five different systems.
>
> Can you build something that does this?
> Thanks,
> Sarah

Methodology

Methodology - Scoring

The Plan is what matters

- Agents were only scored on their ability to **describe a correct plan**.
- Results **weren't penalized** for incorrect or non-compiling code

How results were scored

- A scoring sheet defined the key elements agents needed to articulate
- We used multiple LLMs as judges in a multi-stage review process:
  - Initial LLM scored results against the scoring sheet (GPT-5.1)
  - Second LLM critiqued the initial scores (Gemini-2.5-Pro)
  - Original LLM reviewed critiques and finalized its scoring (GPT-5.1)
- A final LLM collated all feedback and produced the final scores (GPT-5.1)

Judges evaluated **only the plan output** and were **blind to the experimental condition and hypothesis.**

Iterations

- Each scenario was run against an LLM **30 times**, and the scores were averaged

Methodology - Success Criteria

AI Agents were required to provide a plan, which

- Selected the 5 correct API endpoints (path + verb)
- Identify that ID’s didn’t match up, and include ID resolulition calls
- API Calls needed to be ordered correctly
- Describe requires business logic (Pseduocode or plain text description OK)
- Consider failure scenarios, and describe a reasonable strategy for handling an unhappy path

Methodology - Model Selection

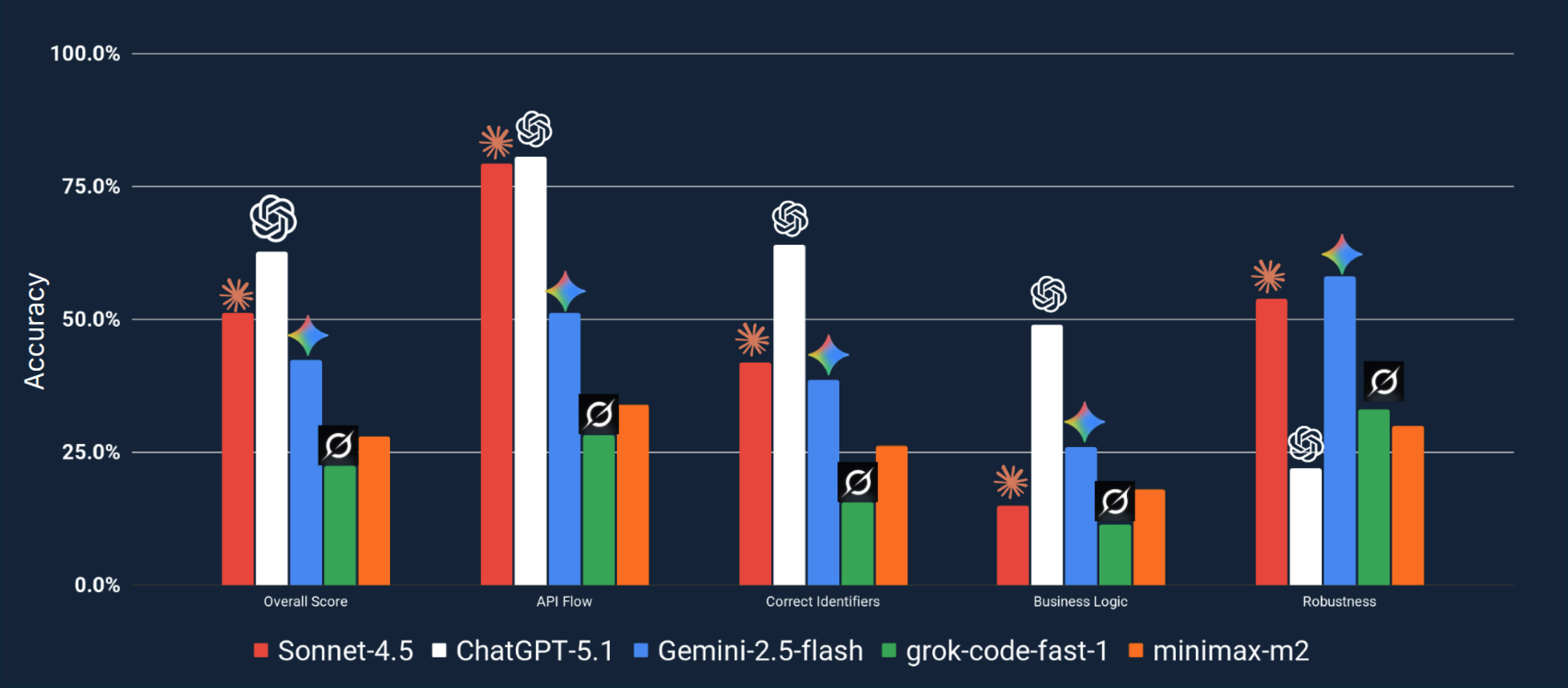
Initial tests were performed on flagship models, as well as the top trending models on OpenRouter for Coding tasks.

Based on performance in the baseline tests, remaining tests were only performed on Sonnet-4.5 and ChatGPT-5.1

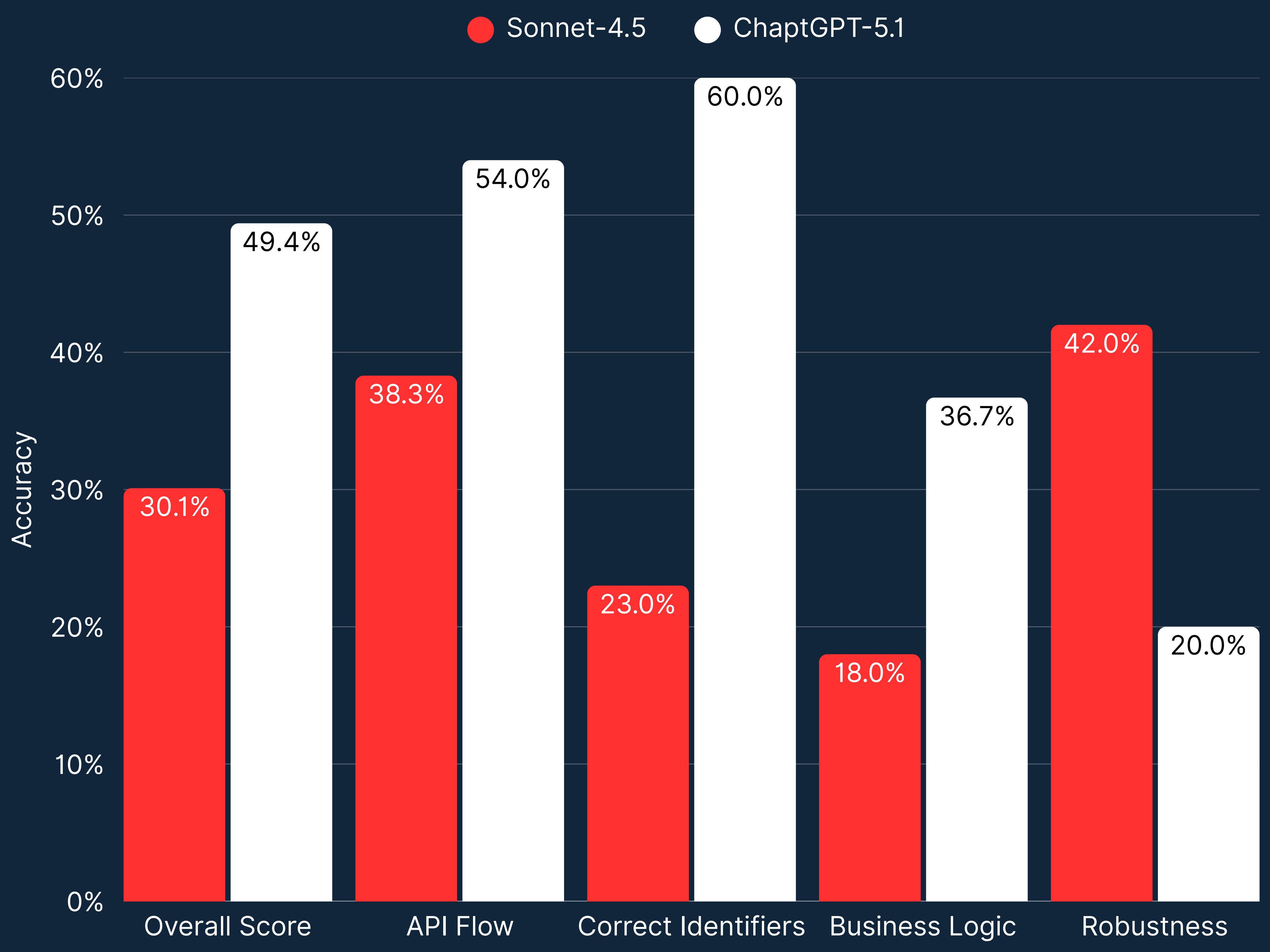
Results

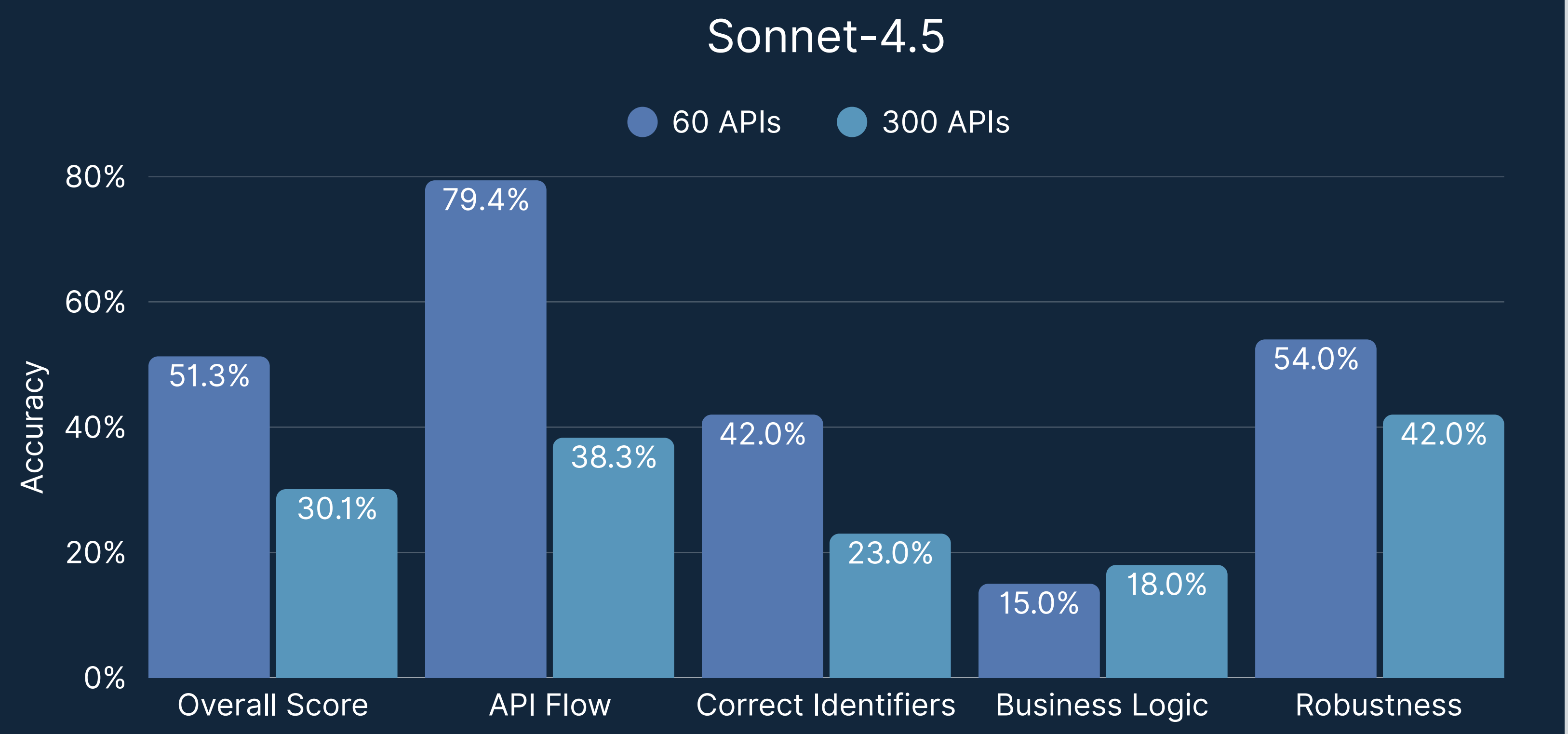
Orchestrate 5 APIs from a population of 60

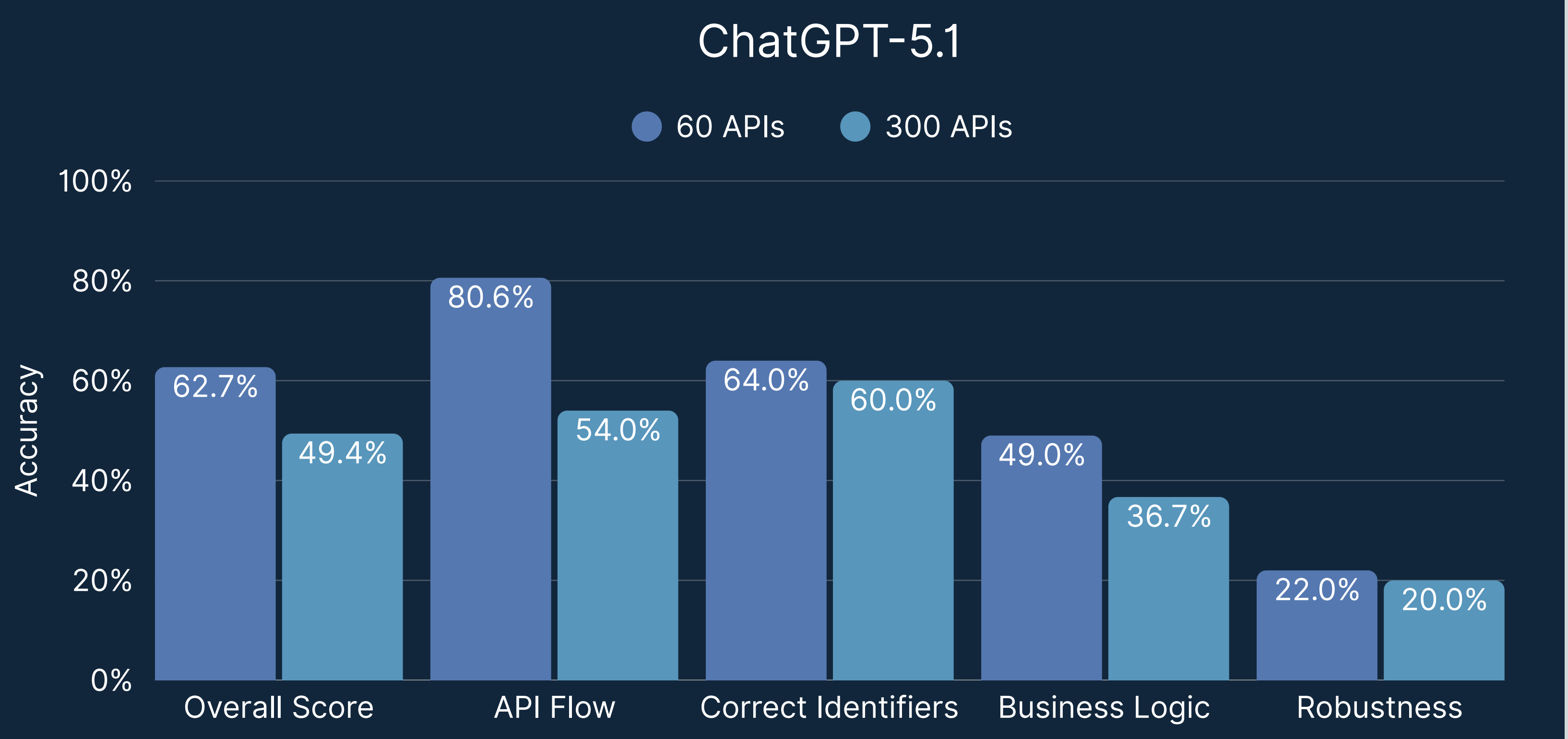
(Baseline)



Orchestrate 5 APIs from a population of 300

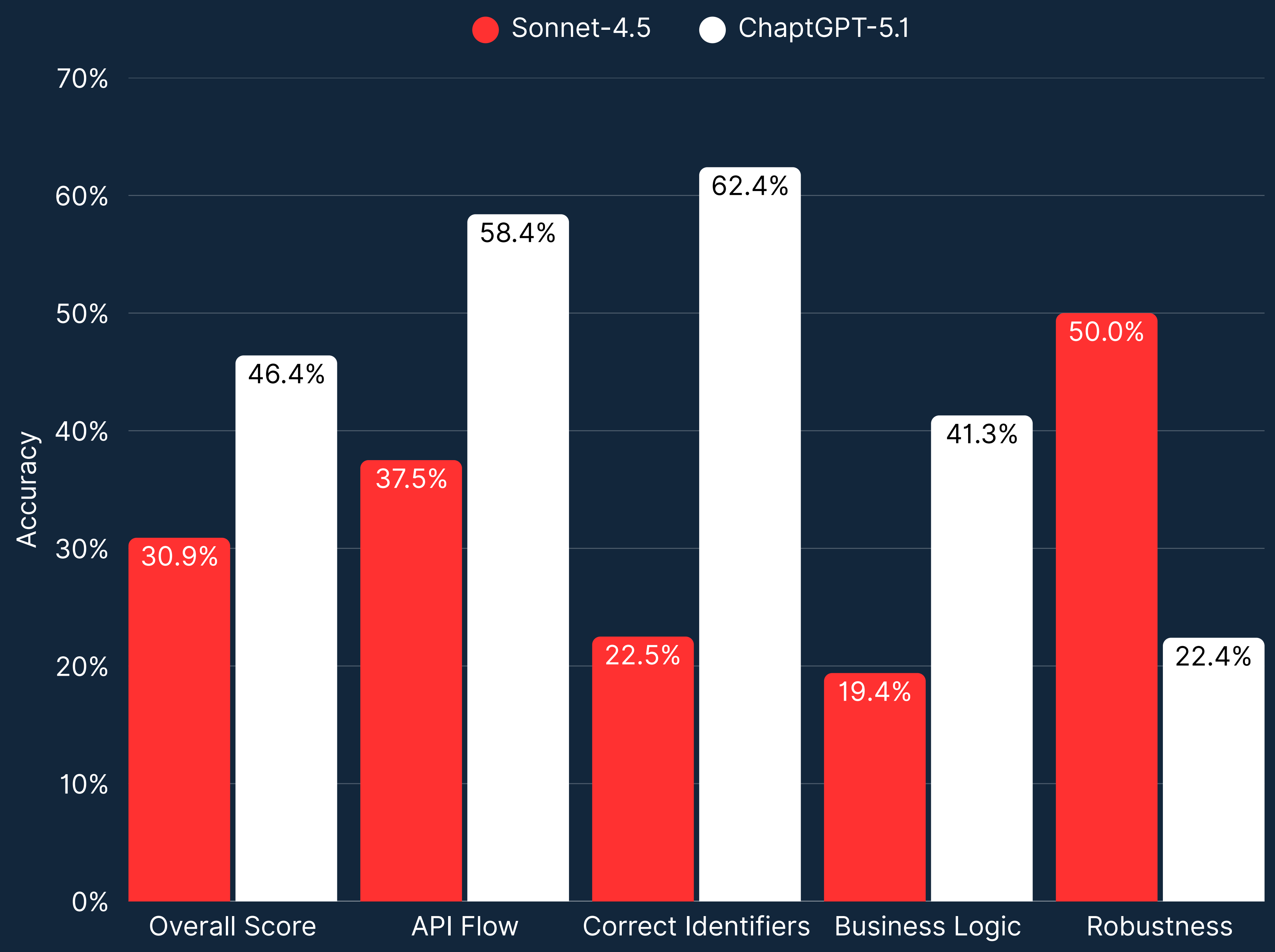


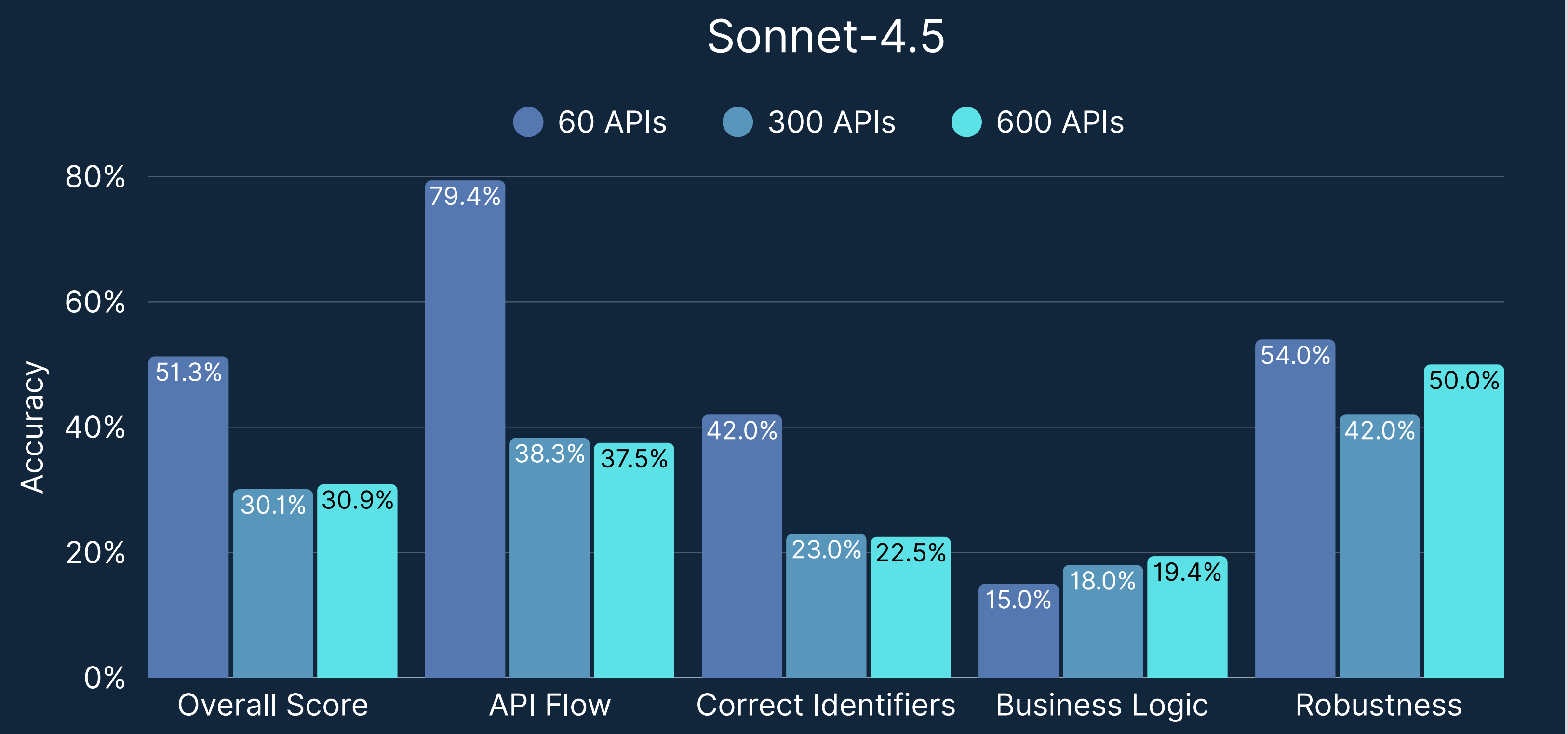


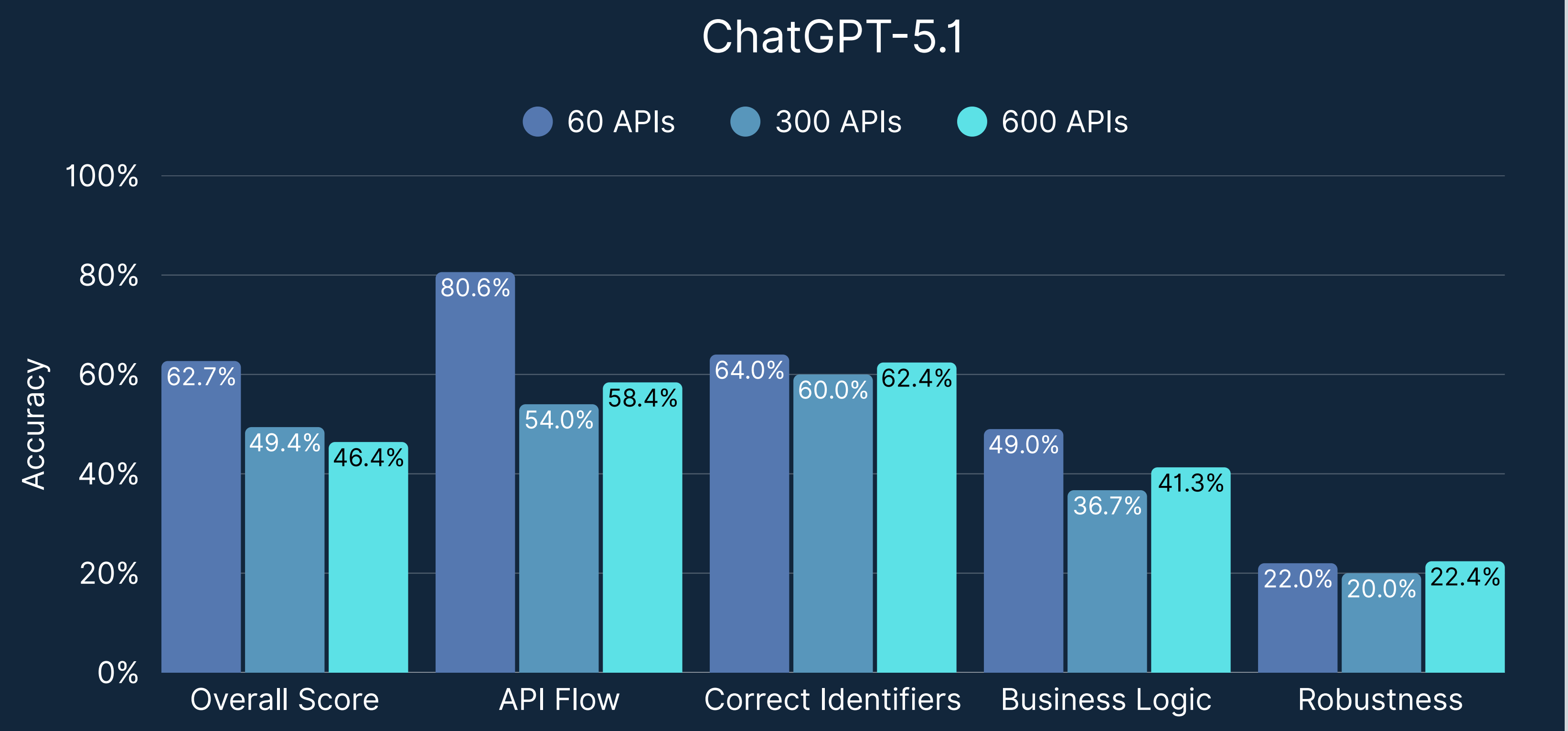


orbital

Orchestrate 5 APIs from a population of 600







orbital

Observations

Flagship models perform adequately only at small scale

With ~60 APIs, both flagship models achieved usable baseline accuracy:

- ChatGPT-5.1: ~49–63% overall, depending on metric
- Sonnet-4.5: ~30–51% overall

Other tested models underperformed at this scale and were excluded from larger-scale tests.

Accuracy drops sharply as API surface area increases from 60 → 300 endpoints

Key planning tasks degraded substantially:

- **Overall accuracy** fell by **~20–30 percentage points**
- **Correct identifier resolution** dropped by **~35–40 percentage points**
- **API flow accuracy** dropped by **~25–40 percentage points**

This decline was consistent across both models.

Observations

Performance stabilises beyond ~300 endpoints.

Increasing the API population from 300 → 600 endpoints resulted in minimal additional degradation (typically ≤5 percentage points across metrics), indicating an early complexity threshold rather than linear decay.

Planning failures are concentrated in reasoning-heavy tasks.

The steepest declines occurred in:

- Selecting the correct APIs
- Resolving identifiers across systems
- Expressing correct business logic

Lower-level robustness metrics were less affected by scale.

Observations

ChatGPT-5.1 consistently outperformed Sonnet-4.5 across planning dimensions.

Across all scales tested, ChatGPT-5.1 scored higher on:

- API flow (by ~15–30 points)
- Correct identifier usage (by ~20–40 points)
- Business logic expression (by ~15–25 points)

The performance gap widens with scale.

Differences between the models were modest at 60 APIs, but **increased materially at 300 and 600 APIs**, suggesting stronger resilience to combinatorial planning complexity in ChatGPT-5.1.

Semantic Layer

Adding a semantic Layer

Semantic metadata was introduced into the OpenAPI specs (using the Taxi OpenAPI format), and scenarios re-tested against 600 API endpoints.

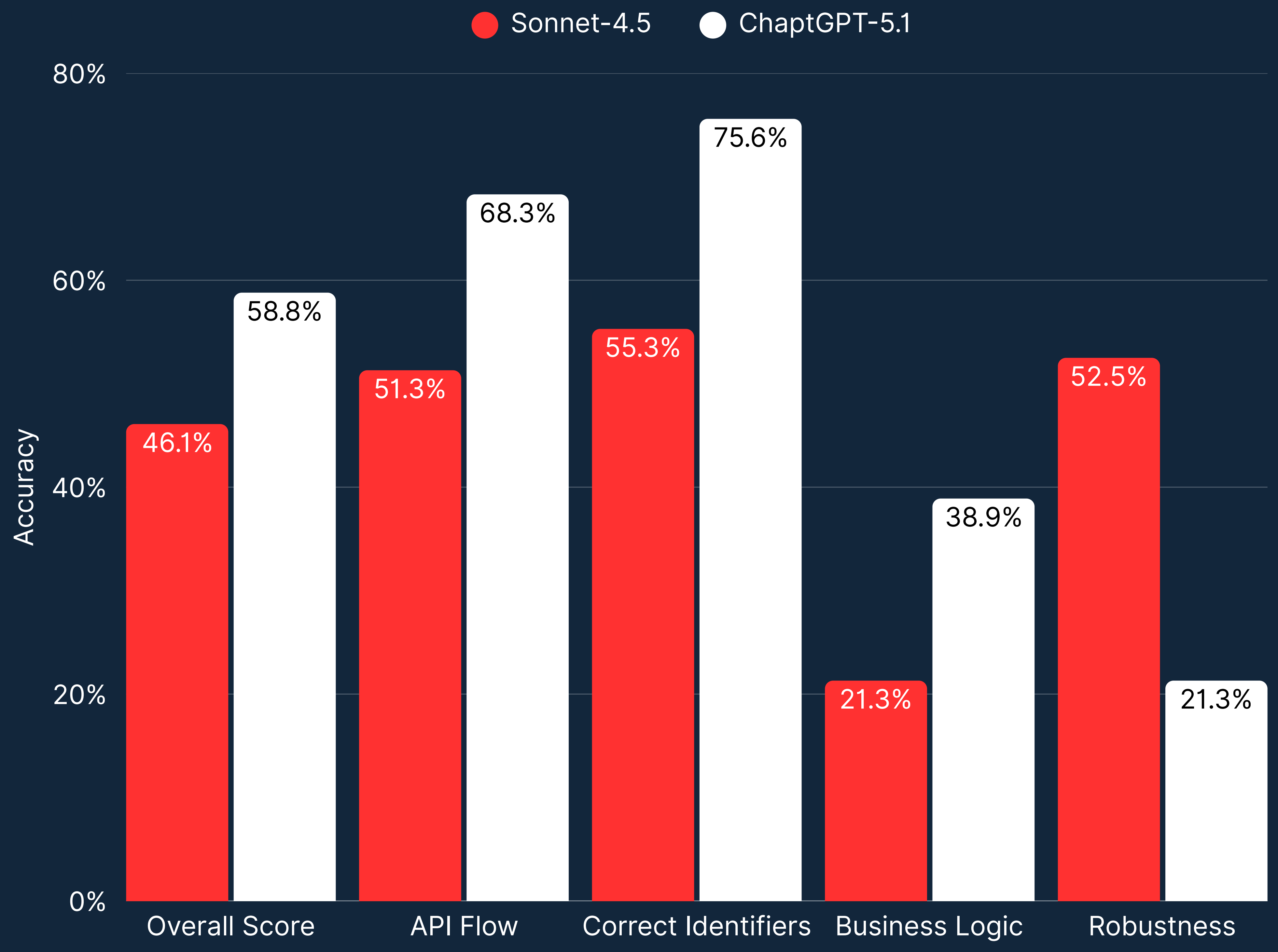
Prompts were updated to instruct the LLMs to:

> *"…consider semantic metadata (expressed via OpenAPI x-taxi-type annotations), which explicitly describe the semantic meaning of data contained within each field."*

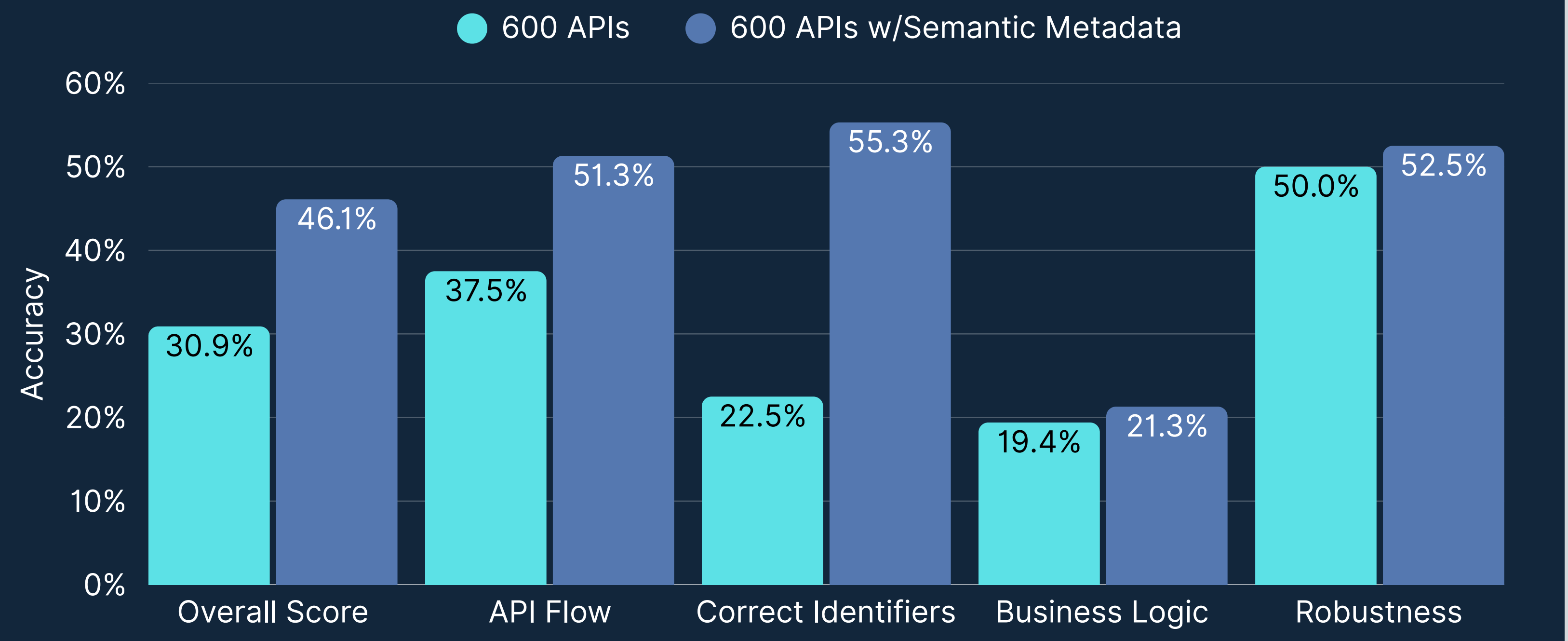
No other changes were made to the experiment setup.

Notably, the LLMs were **not given any additional explanation or training** on semantic metadata, Taxi, TaxiQL, or the Taxi format beyond what was already present through their base training.

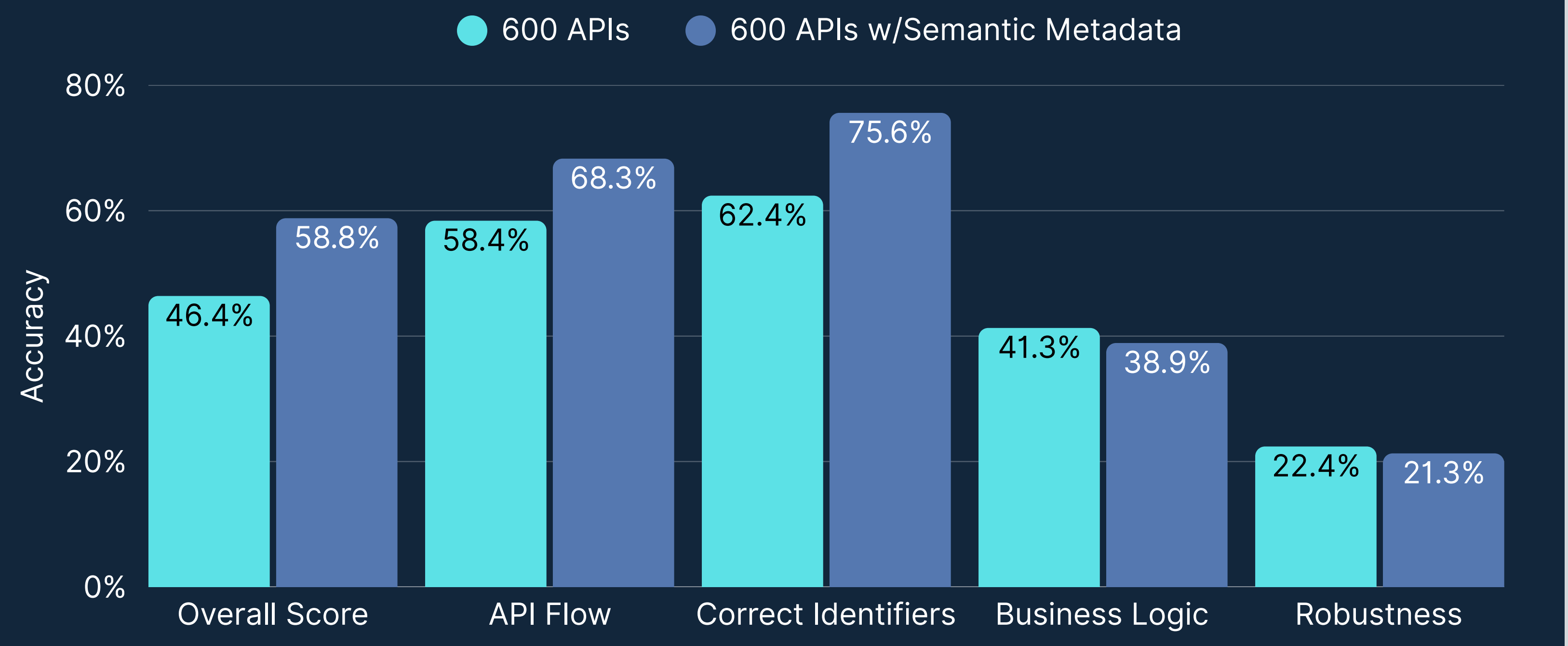
Orchestrate 5 APIs from a population of 600, with semantic metadata in OpenAPI specs



Sonnet-4.5



ChatGPT-5.1



Observations

Adding semantic metadata materially improves planning accuracy

Semantic metadata produced consistent accuracy gains across all planning dimensions.

When re-tested against 600 APIs, both models showed improvements in:

- Overall planning accuracy **(+12–15 percentage points)**
- API flow selection **(+13–16 points)**
- Identifier resolution **(+30–33 points)**

Relative improvements were substantial despite no model or prompt tuning.

Improvements are attributable to added semantic signal, not the specific annotation format.

Key Takeaway

Even minimal semantic markup significantly reduces planning ambiguity for LLMs operating at scale.

Declarative Orchestration Language

Using TaxiQL to express orchestration requirements

orbital

Using TaxiQL to express orchestration requirements

Prompts were updated to instruct the LLMs to:

> *"Leveraging Taxi metadata, use the API query language TaxiQL to express data fetching requirements. Where possible, defer API orchestration to the TaxiQL execution layer."*

Additionally, schemas were presented in Taxi format, rather than OpenAPI.

The LLMs were not provided with any additional explanation, fine-tuning, or training on Taxi or TaxiQL beyond what was already present in their base training.

As with all tests, TaxiQL output was evaluated **as a planning artefact**, not as executable code. Responses were scored on whether they represented a **valid and complete expression of data requirements**; outputs were **not penalised** if the TaxiQL did not compile.

TaxiQL compilation

We tracked TaxiQL syntactic correctness separately to maintain fair test conditions.

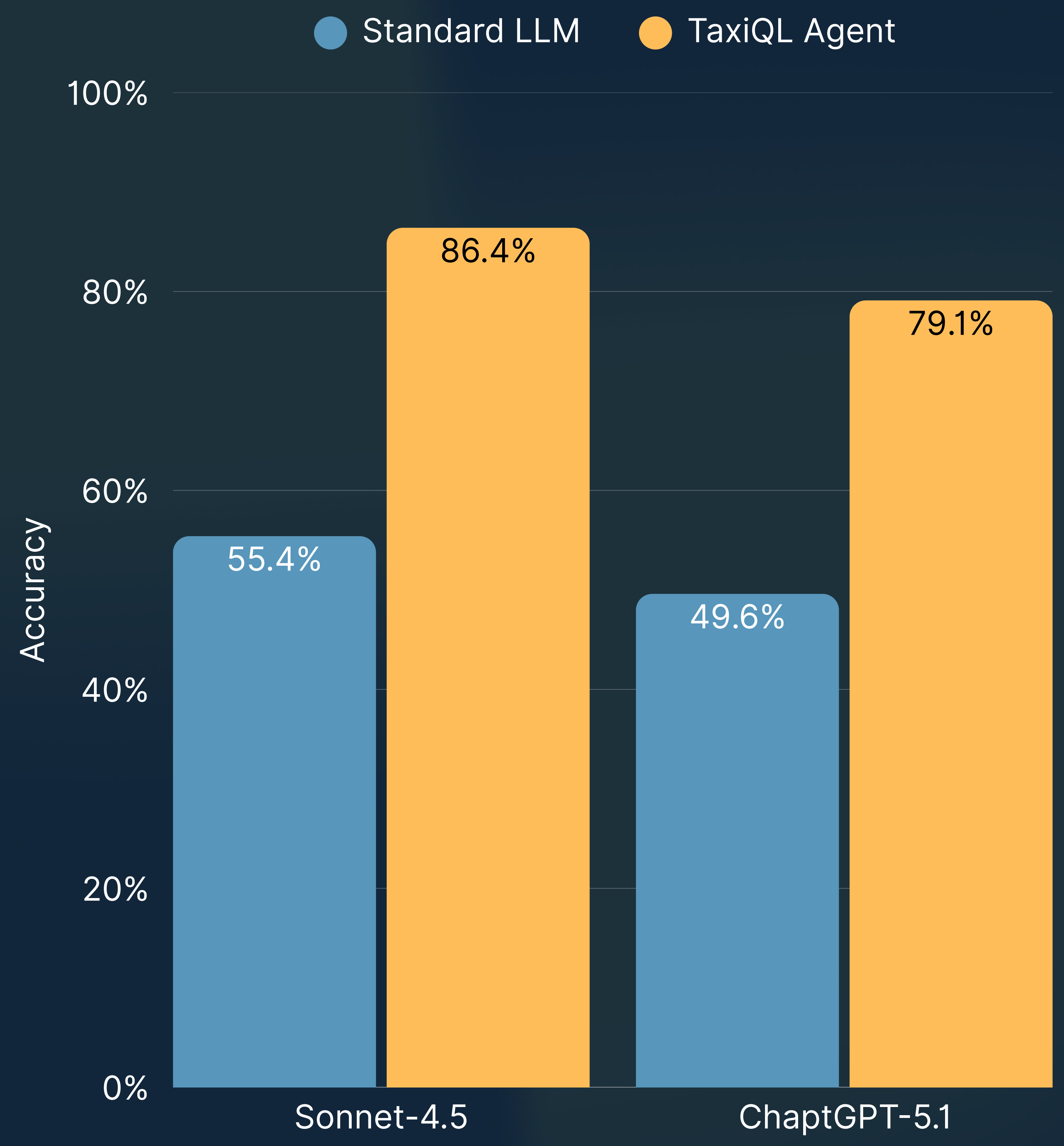
All test scenarios used the same base LLMs (Sonnet-4.5, ChatGPT-5.1) without specialized code generation tooling. This ensured consistent comparisons.

Our scoring measured planning intent - requesting the correct data, ordering, and resolving identifiers - not syntactic perfection. Introducing a TaxiQL-specific coding agent would have skewed comparisons in favor of TaxiQL.

When we tested a specialized coding agent designed for TaxiQL generation, compilation rates improved substantially, demonstrating that syntactic correctness is achievable with appropriate tooling.

These rates are shown for transparency and do not factor into the study's primary findings

TaxiQL syntactic correctness
(reference only - not scored in primary results)

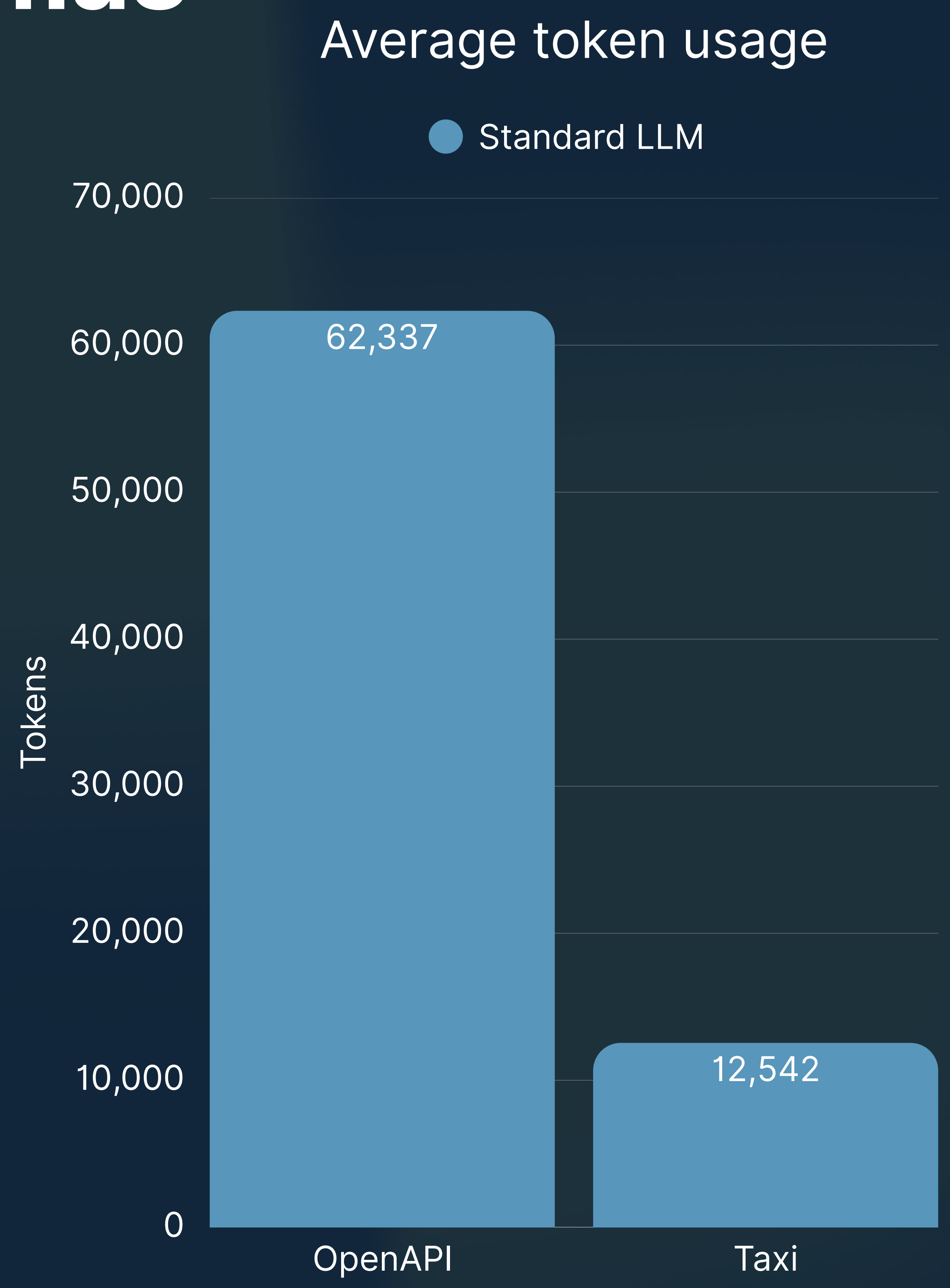


Taxi vs OpenAPI for expressing schemas

In this test scenario, API specifications were provided in **Taxi format** rather than OpenAPI.

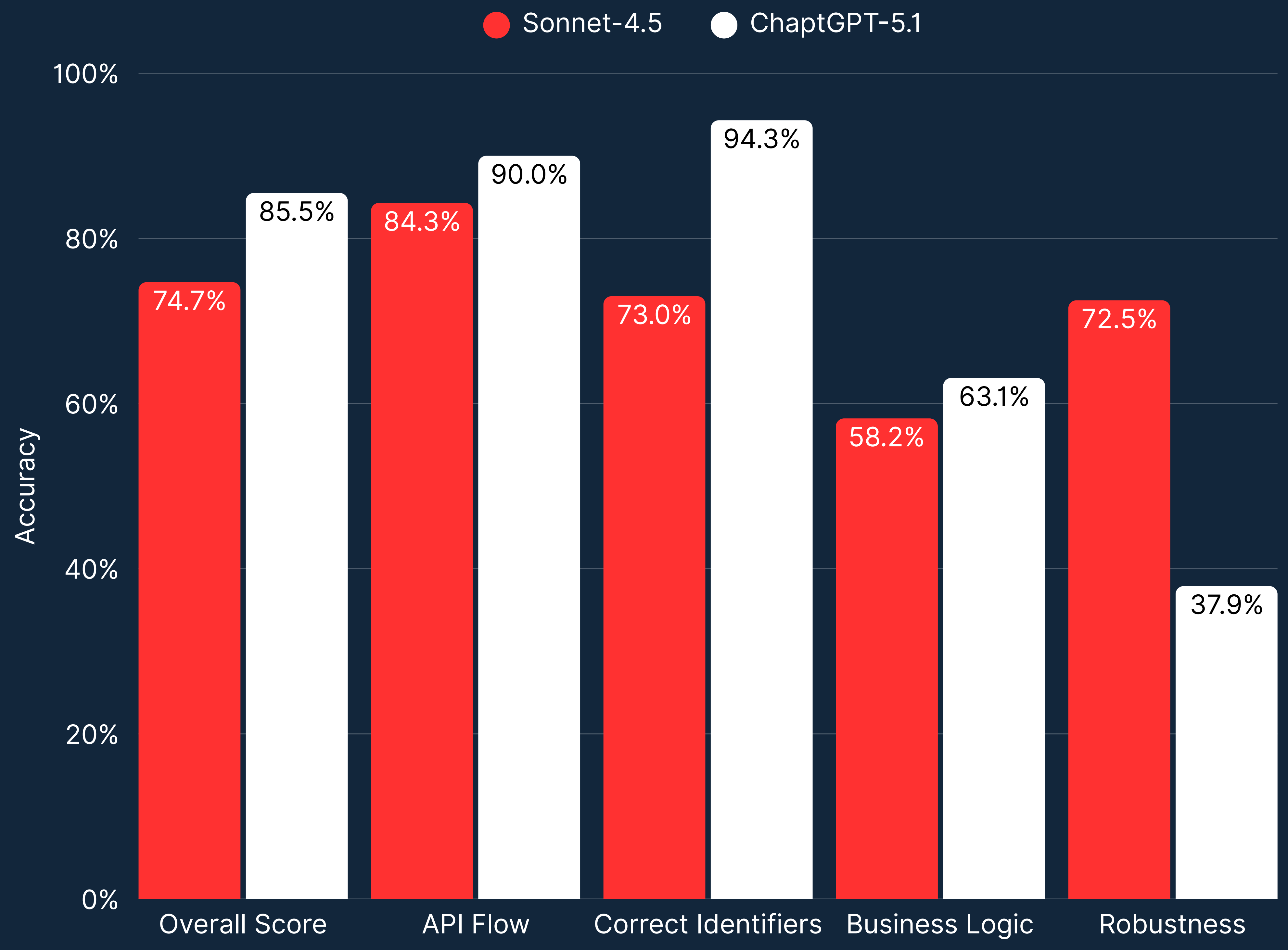
Taxi - a dedicated schema language - required ~80% fewer tokens to represent the same schemas, resulting in a substantially smaller prompt context.

In addition, because Taxi represents **semantic relationships as first-class constructs**, the remaining tokens carried **higher contextual density**, improving the signal-to-noise ratio available to the model.

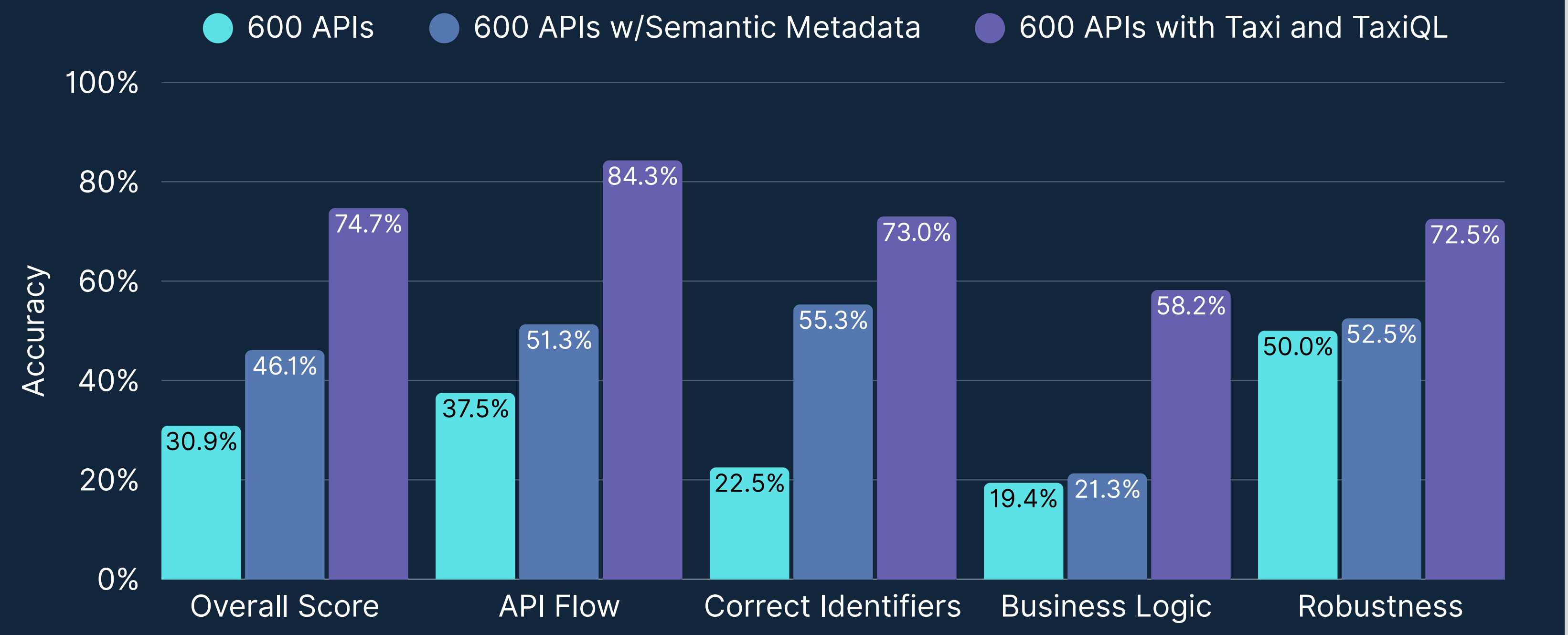


orbital

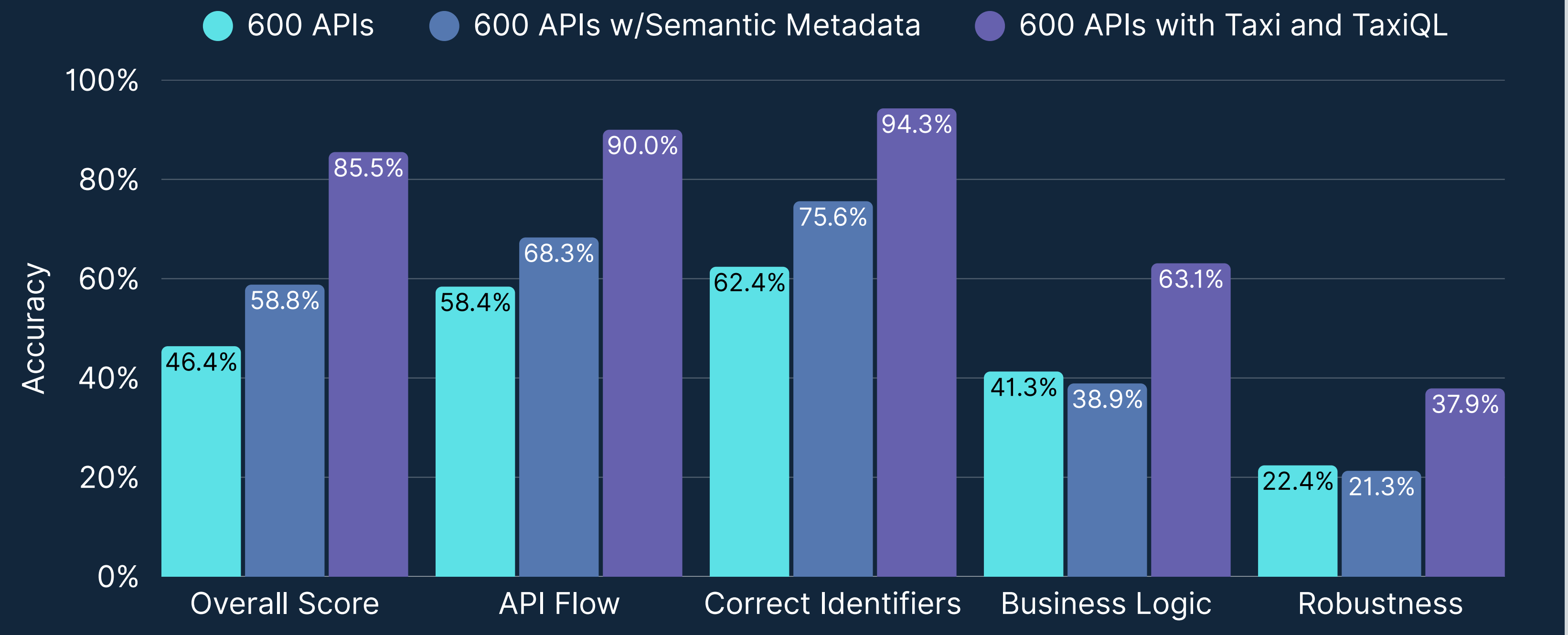
Orchestrate 5 APIs from a population of 600, using Taxi and TaxiQL



Sonnet-4.5



ChatGPT-5.1



orbital

Observations

Declarative orchestration dramatically improves planning accuracy

Expressing orchestration requirements declaratively produced the largest accuracy gains observed in the study.

When planning was expressed using TaxiQL (with schemas provided in Taxi format), overall planning accuracy increased by:

- **Sonnet-4.5:** 30.0% → 74.7% (**~142% relative increase)**
- **ChatGPT-5.1:** 46.4% → 80.5% **(~73% relative increase)**

relative to the unassisted baseline.

Improvements were strongest in reasoning-heavy tasks.

Compared to semantic metadata alone, TaxiQL produced substantial additional gains in:

- Correct identifier resolution (up to +50 percentage points)
- API flow planning (up to +45 points)
- Business logic expression (up to +35 points)

Observations

Declarative orchestration dramatically improves planning accuracy

Gains were achieved without additional model training or fine-tuning.
The LLMs received no TaxiQL-specific instruction beyond prompt guidance; improvements arise from structural changes in how requirements are expressed, not from model adaptation.

This indicates that even though TaxiQL remains a niche language, it’s been around long enough that it has already been scraped and learnt by LLM base training, sufficiently to express requirements in a planning scenario

Accuracy gains compound with reduced context size.
Taxi + TaxiQL simultaneously:

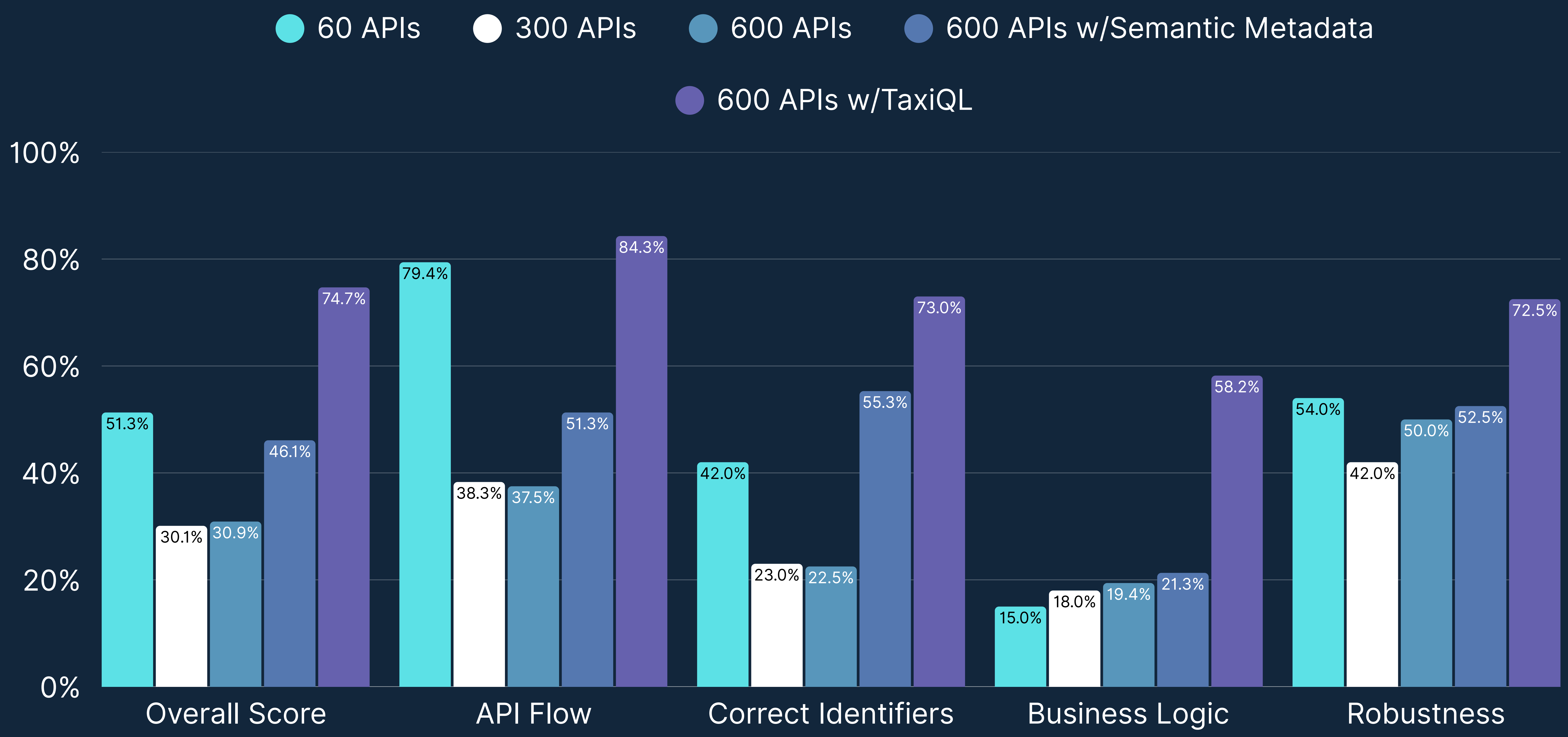
- reduced prompt size by ~80%, and
- increased semantic density per token,
- improving both planning reliability and cost efficiency.

Key Takeaway

**Using a semantic, declarative layer (Taxi + TaxiQL) increased planning accuracy by 73–142%,
while reducing token usage by up to 80%.**

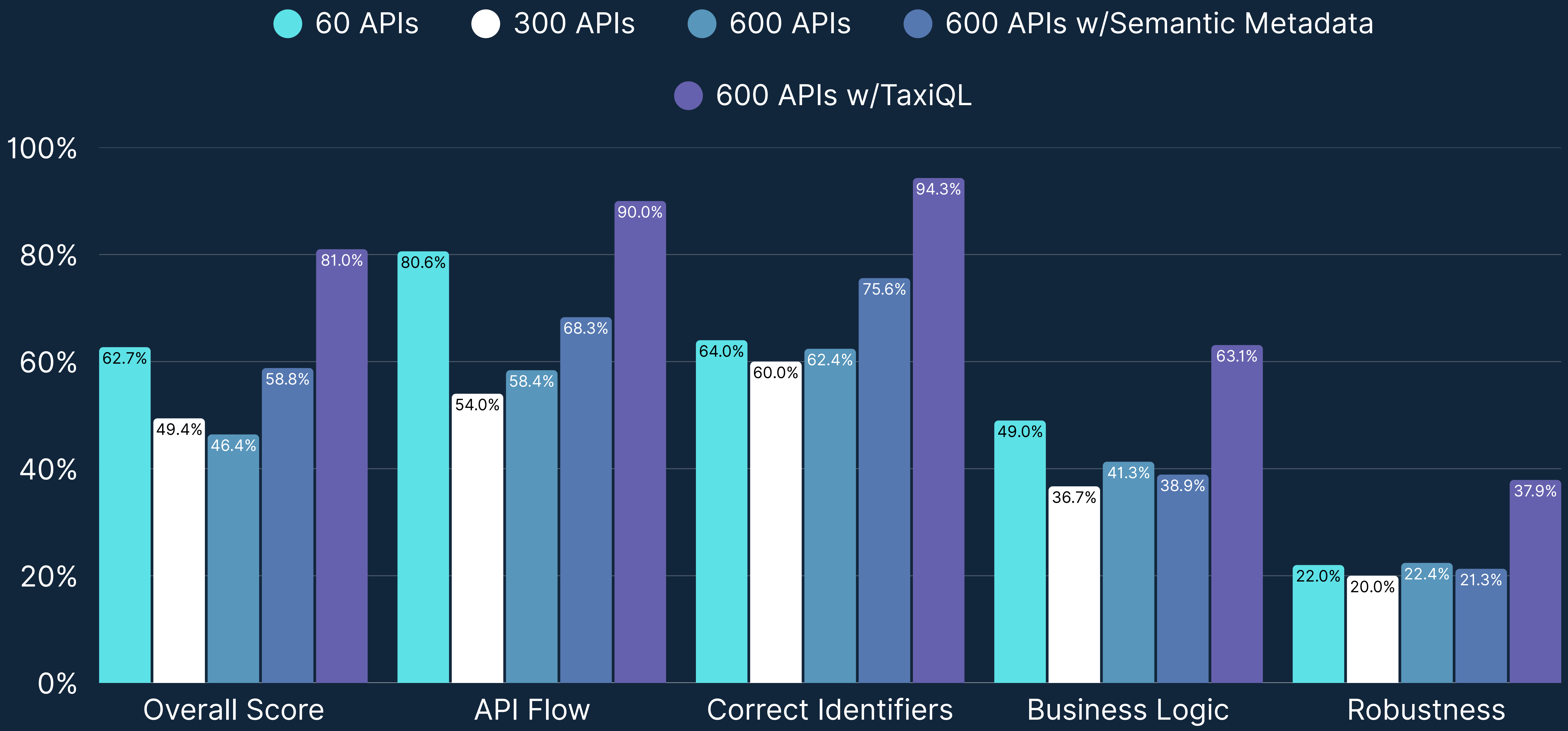
Overall performance

Performance by scenario - Sonnet 4.5



orbital

Performance by scenario - ChatGPT-5.1



orbital

orbital

hello@orbitalhq.com

orbitalhq.com