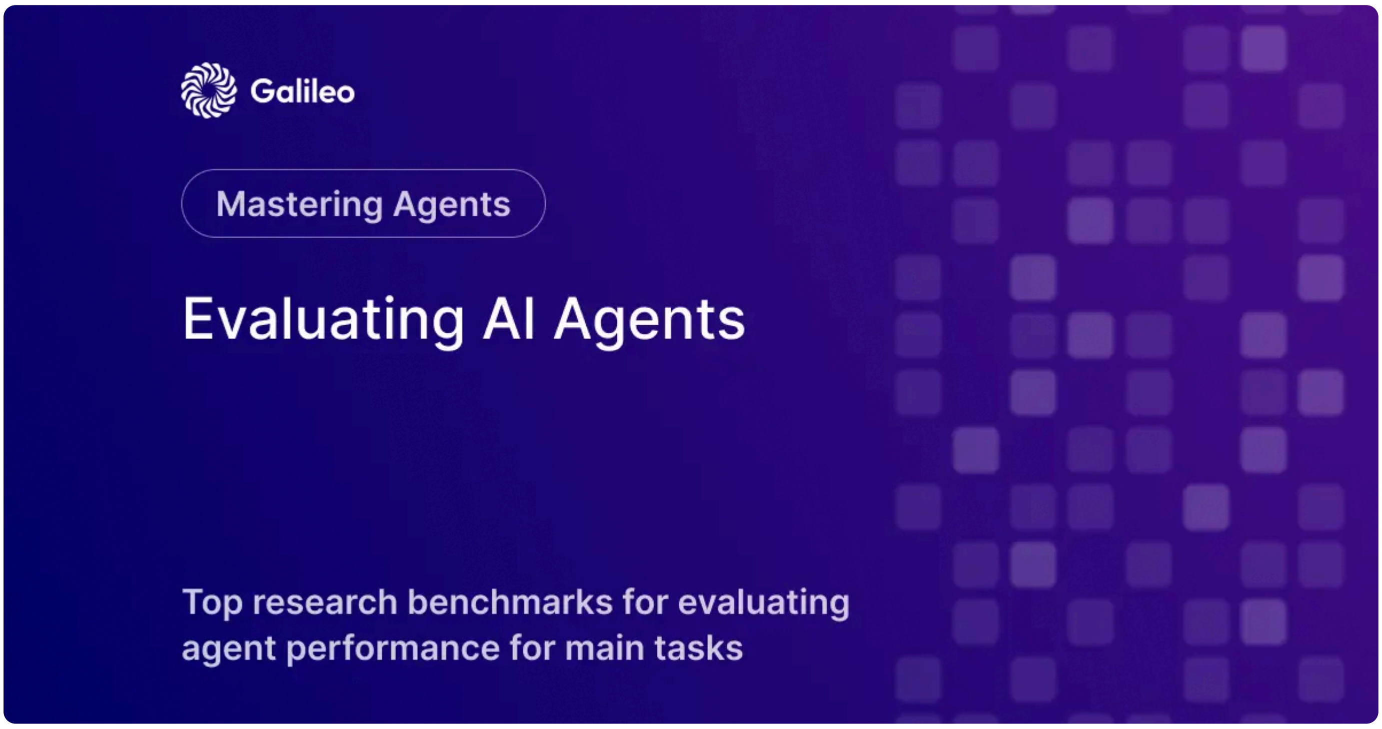
Upc

Mastering Agents: Evaluating AI Agents

Pratik Bhavsar
Galileo Labs


Galileo
Mastering Agents
Evaluating AI Agents
Top research benchmarks for evaluating agent performance for main tasks

14 min read December 18 2024

Table of contents

Show

Imagine you're working with an AI assistant that claims it can help you complete your tasks. Can you trust it to analyze data effectively? To write important press releases? To make complex product decisions?

Hi there! What can I help you with?

Evaluating AI agents isn't like testing traditional so the output matches expected results. These agents perform complex tasks th often have multiple valid approaches. They need to understand context anc

Upc

these systems are both capable and reliable.

In this post we'll explore how researchers are tackling these challenges by examining fundamental capabilities that define an effective AI agent. Each capability requires its own specialized evaluation frameworks. Understanding them helps us grasp both the current state of AI technology and where improvements are needed.

This blog will provide solid fundamentals of evaluating agents that are becoming part of our world.

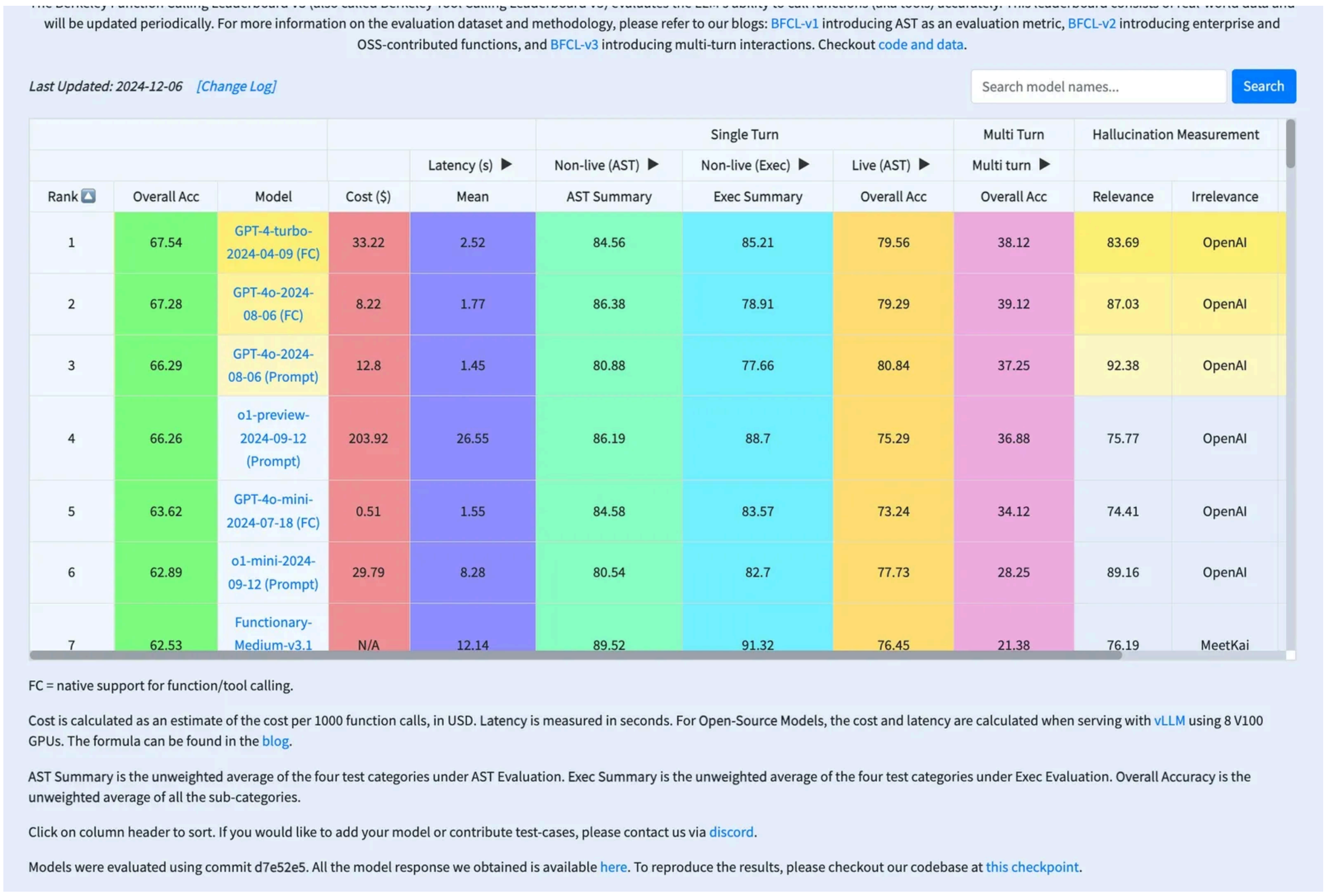
Tool Calling

Let's start with the most important skillet. The ability to select and use appropriate tools has become a cornerstone of AI agent functionality. Ask anyone *what is agent* and they will immediately mention tool calling.

Berkeley Function-Calling Leaderboard (BFCL)

Hi there! What can I help you with?

Upc



will be updated periodically. For more information on the evaluation dataset and methodology, please refer to our blogs: BFCL-v1 introducing AST as an evaluation metric, BFCL-v2 introducing enterprise and OSS-contributed functions, and BFCL-v3 introducing multi-turn interactions. Checkout code and data.

Last Updated: 2024-12-06 *[Change Log]*

Search model names... Search

					Single Turn			Multi Turn	Hallucination Measurement	
				Latency (s) ▶	Non-live (AST) ▶	Non-live (Exec) ▶	Live (AST) ▶	Multi turn ▶		
Rank	Overall Acc	Model	Cost ($)	Mean	AST Summary	Exec Summary	Overall Acc	Overall Acc	Relevance	Irrelevance
1	67.54	GPT-4-turbo-2024-04-09 (FC)	33.22	2.52	84.56	85.21	79.56	38.12	83.69	OpenAI
2	67.28	GPT-4o-2024-08-06 (FC)	8.22	1.77	86.38	78.91	79.29	39.12	87.03	OpenAI
3	66.29	GPT-4o-2024-08-06 (Prompt)	12.8	1.45	80.88	77.66	80.84	37.25	92.38	OpenAI
4	66.26	o1-preview-2024-09-12 (Prompt)	203.92	26.55	86.19	88.7	75.29	36.88	75.77	OpenAI
5	63.62	GPT-4o-mini-2024-07-18 (FC)	0.51	1.55	84.58	83.57	73.24	34.12	74.41	OpenAI
6	62.89	o1-mini-2024-09-12 (Prompt)	29.79	8.28	80.54	82.7	77.73	28.25	89.16	OpenAI
7	62.53	Functionary-Medium-v3.1	N/A	12.14	89.52	91.32	76.45	21.38	76.19	MeetKai

FC = native support for function/tool calling.

Cost is calculated as an estimate of the cost per 1000 function calls, in USD. Latency is measured in seconds. For Open-Source Models, the cost and latency are calculated when serving with vLLM using 8 V100 GPUs. The formula can be found in the blog.

AST Summary is the unweighted average of the four test categories under AST Evaluation. Exec Summary is the unweighted average of the four test categories under Exec Evaluation. Overall Accuracy is the unweighted average of all the sub-categories.

Click on column header to sort. If you would like to add your model or contribute test-cases, please contact us via discord.

Models were evaluated using commit d7e52e5. All the model response we obtained is available here. To reproduce the results, please checkout our codebase at this checkpoint.

University of California, Berkeley has pioneered a comprehensive framework [Berkeley Function Calling Leaderboard](#) (BFCL) for evaluating these capabilities. It has evolved through multiple versions to address increasingly sophisticated aspects of function calling.

The journey began with **BFCL v1**, which established the foundation for evaluating function-calling capabilities. This initial version introduced a diverse evaluation dataset question-function-answer pairs covering multiple programming languages including Python, Java, JavaScript, and REST APIs.

The framework also evaluated complex scenarios where agents needed to select one or more functions from multiple options, or make parallel function calls together. This work revealed significant insights into how different models handled tool selection, with proprietary models like GPT-4 leading in performance, followed closely by open-source alte

Hi there! What can I help you with?

BFCL v2 introduced real-world complexity throug

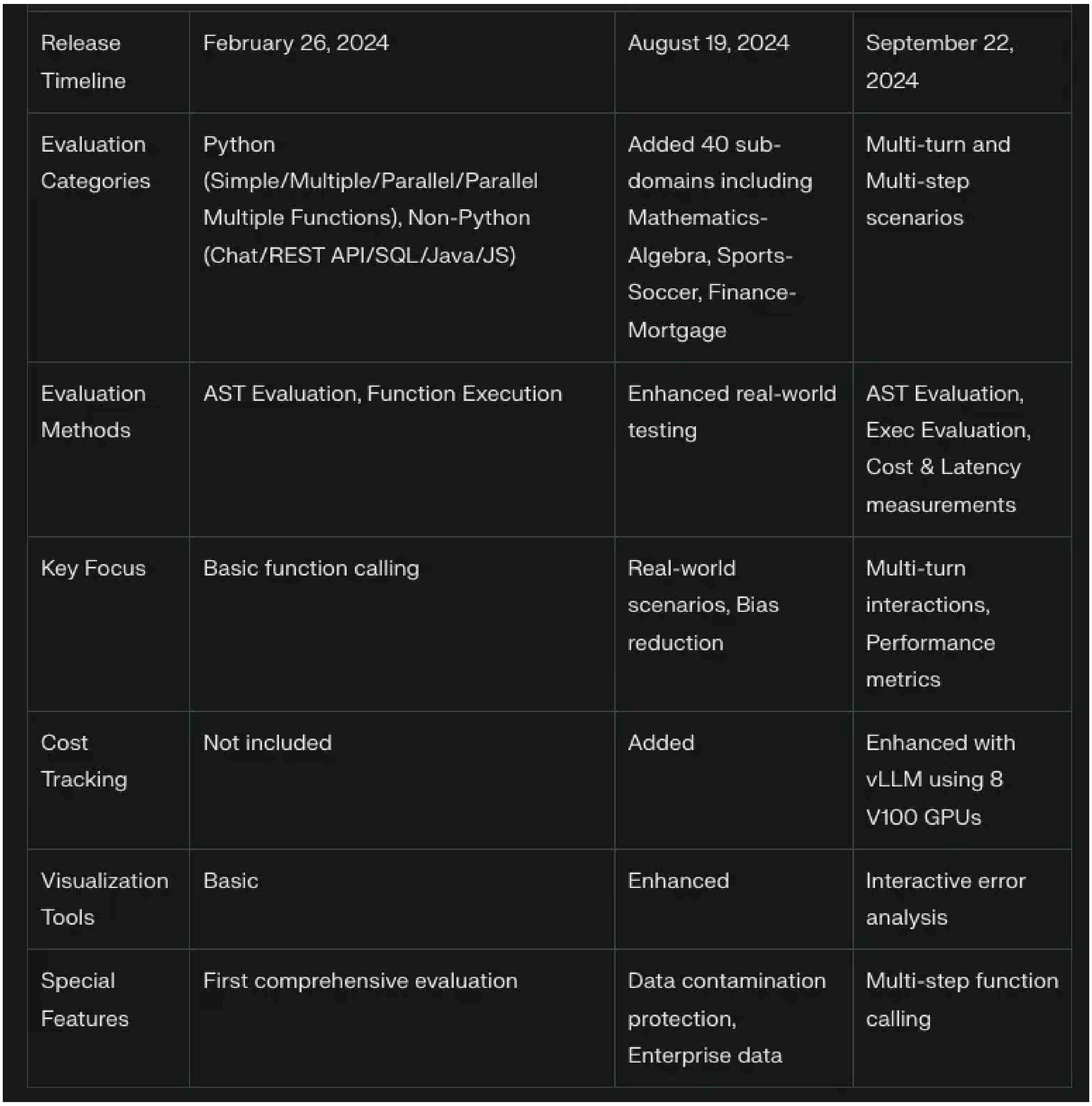
version addressed crucial issues like bias and data contamination while focusing on dynamic, real-world scenarios. The evaluation expanded to include more sophisticated test cases. It revealed that in real-world usage, there's a higher

Upc

The latest iteration, **BFCL v3** pushed the boundaries further by introducing **multi-turn** and **multi-step** evaluation scenarios. This version recognized that real-world applications often require complex sequences of interactions, where agents must maintain context, handle state changes, and adapt their strategies based on previous outcomes. It introduced sophisticated evaluation metrics including state-based evaluation, which examines how well agents maintain and modify system state, and response-based assessment, which analyzes the appropriateness and efficiency of function selection.

Hi there! What can I help you with?

Upc



Release Timeline	February 26, 2024	August 19, 2024	September 22, 2024
Evaluation Categories	Python (Simple/Multiple/Parallel/Parallel Multiple Functions), Non-Python (Chat/REST API/SQL/Java/JS)	Added 40 sub-domains including Mathematics-Algebra, Sports-Soccer, Finance-Mortgage	Multi-turn and Multi-step scenarios
Evaluation Methods	AST Evaluation, Function Execution	Enhanced real-world testing	AST Evaluation, Exec Evaluation, Cost & Latency measurements
Key Focus	Basic function calling	Real-world scenarios, Bias reduction	Multi-turn interactions, Performance metrics
Cost Tracking	Not included	Added	Enhanced with vLLM using 8 V100 GPUs
Visualization Tools	Basic	Enhanced	Interactive error analysis
Special Features	First comprehensive evaluation	Data contamination protection, Enterprise data	Multi-step function calling

Different versions of Berkeley Function Calling Leaderboard

Insights from Failures

Through these iterations, several critical challenges in tool selection have emerged. One persistent issue is **implicit action recognition**. Agents often struggle to identify necessary preliminary steps th user requests. For instance, an agent might attem checking if it exists, or try to post content without verifying authentication status. State management presents another significant challenge, with agents

Hi there! What can I help you with?

Upc

The evaluation framework revealed interesting patterns in how different models handle tool selection. While both proprietary and open-source models perform similarly in simple function calling scenarios, more complex situations involving multiple or parallel function calls tend to showcase larger performance gaps. This insight has proven valuable for understanding where current models excel and where they need improvement.

T-bench

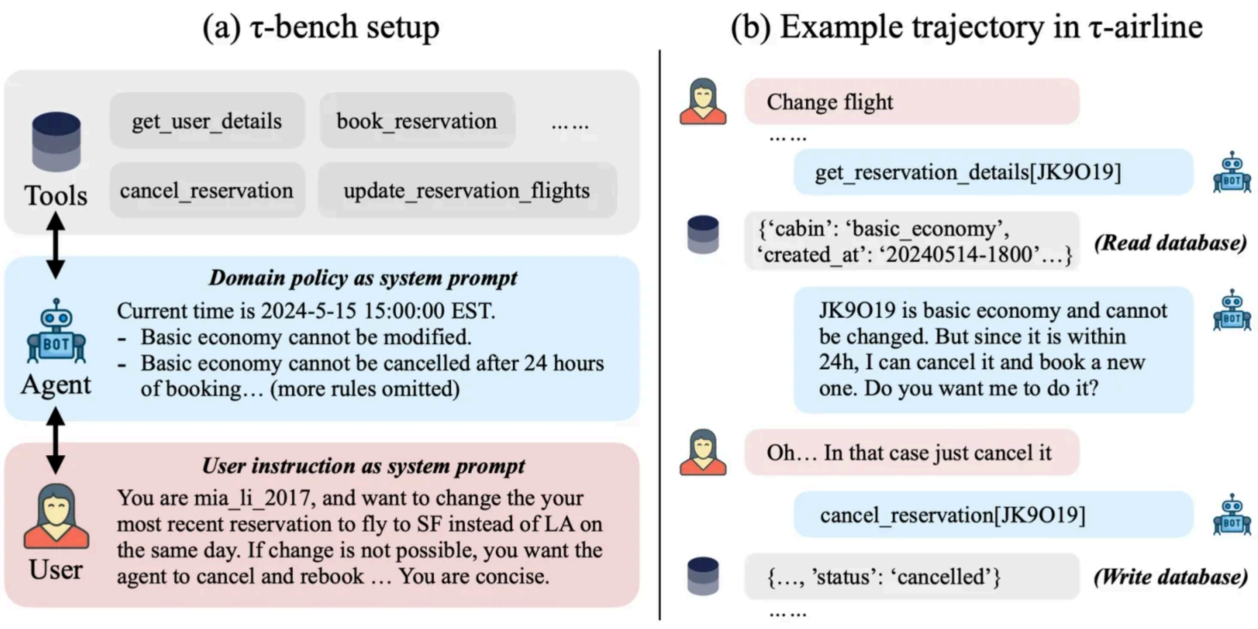

(a) τ-bench setup
Tools
get_user_details
book_reservation
……
cancel_reservation
update_reservation_flights
Agent
Domain policy as system prompt
Current time is 2024-5-15 15:00:00 EST.
- Basic economy cannot be modified.
- Basic economy cannot be cancelled after 24 hours of booking… (more rules omitted)
User
User instruction as system prompt
You are mia_li_2017, and want to change the your most recent reservation to fly to SF instead of LA on the same day. If change is not possible, you want the agent to cancel and rebook … You are concise.
(b) Example trajectory in τ-airline
Change flight
……
get_reservation_details[JK9O19]
{'cabin': 'basic_economy', 'created_at': '20240514-1800'…}
(Read database)
JK9O19 is basic economy and cannot be changed. But since it is within 24h, I can cancel it and book a new one. Do you want me to do it?
Oh… In that case just cancel it
cancel_reservation[JK9O19]
{…, 'status': 'cancelled'}
(Write database)
……

Figure 1: (a) In τ-bench, an agent interacts with database API tools and an **LM-simulated user** to complete tasks. The benchmark tests an agent's ability to collate and convey all required information from/to users through multiple interactions, and solve complex issues on the fly while ensuring it **follows guidelines** laid out in a domain-specific policy document. (b) An example trajectory in τ-airline, where an agent needs to reject the user request (change a basic economy flight) following domain policies and propose a new solution (cancel and rebook). This challenges the agent in long-context zero-shot reasoning over complex databases, rules, and user intents.

Taking this evaluation framework further, the recently introduced τ-bench represents a significant advancement in tool selection assessment by focusing on **real-world interactions between agents and users**. Unlike previous benchmarks that typically evaluated agents on single-step interactions with pre-defined inputs, τ-bench creates a more realistic environme in dynamic conversations while following specific c

Hi there! What can I help you with?

The benchmark's innovation lies in its three-layered approach to evaluation. First, it provides agents with access to **realistic databases and APIs** that mirror real-

simulate human users, creating natural, varied interactions that test an agent's ability to gather information and respond appropriately over multiple turns.

τ-bench specifically focuses on two domains that represent common real-world applications: **retail customer service** (τ-retail) and **airline reservations** (τ-airline). In the retail domain, agents must handle tasks like order modifications, returns, and exchanges while adhering to specific store policies. The airline domain presents even more complex challenges, requiring agents to navigate flight bookings, changes, and cancellations while considering various rules about fares, baggage allowances, and membership tiers.

Instead of simply checking if an agent generates the correct function calls, it evaluates the final state of the database after the interaction. This approach acknowledges that there might be multiple valid paths to achieve the same goal, better reflecting real-world scenarios. Additionally, the benchmark introduces a new metric called **pass^k** which measures an agent's consistency across multiple attempts at the same task.

Insights from Failures

The results from τ-bench have been both enlightening and sobering. Even state-of-the-art models like GPT-4 succeed on less than 50% of tasks, with performance particularly poor on the more complex airline domain (around 35% success rate). More concerning is the dramatic drop in performance when consistency is required across multiple attempts. In retail scenarios, while GPT-4 achieves a 61% success rate on single attempts, this drops below 25% when requiring consistent performance across eight attempts at the same task. This reveals a critical gap between current capabilities and the reliability needed for real-world deployment.

Through detailed analysis τ-bench has identified s

Hi there! What can I help you with?

current tool selection capabilities. A significant por
55%) stem from agents either **providing incorrect information or making wrong arguments** in function calls. These errors often occur when agents need to **reason over complex databases or handle numerical calculations**. For

Upc

Another quarter of failures result from incorrect decision-making, particularly in **understanding and following domain-specific rules**. This becomes especially apparent in the airline domain, where removing the policy document from GPT-4's context leads to a dramatic 22.4% drop in performance. This suggests that even advanced models struggle to consistently apply complex domain rules, such as those governing baggage allowances for different membership tiers and cabin classes.

The benchmark also revealed significant challenges in handling compound requests. When tasks require multiple database writes or involve several user requests, performance degrades substantially. This points to limitations in agents' ability to maintain context and systematically address all aspects of a complex request. For example, agents might successfully modify one order but fail to apply the same changes to other relevant orders in the same conversation.

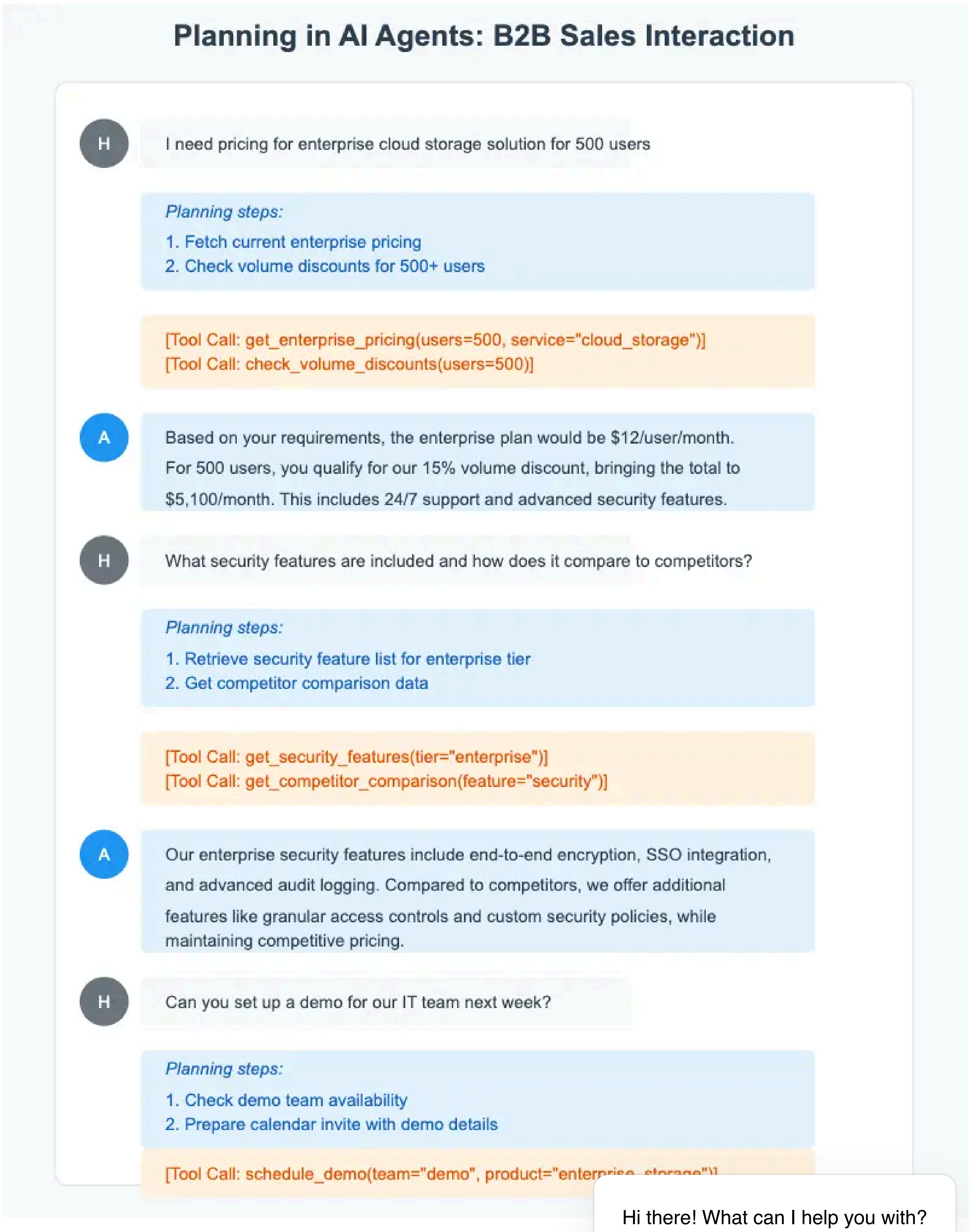
Looking ahead, τ-bench has highlighted several critical areas for improvement in tool selection capabilities. Future research needs to focus on:

- Enhancing agents' ability to reason over complex databases and maintain numerical accuracy
- Developing better methods for understanding and consistently applying domain-specific rules
- Improving long-term context maintenance in multi-step interactions
- Creating more robust approaches to handling compound requests
- Building systems that can maintain consistent performance across multiple interactions with different users

The gap between current tool calling capabilities a suggests that significant advances are still needed systems can be reliably deployed in critical customer-facing roles.

Hi there! What can I help you with?

Upc


Planning in AI Agents: B2B Sales Interaction

H: I need pricing for enterprise cloud storage solution for 500 users

Planning steps:
1. Fetch current enterprise pricing
2. Check volume discounts for 500+ users

[Tool Call: get_enterprise_pricing(users=500, service="cloud_storage")]
[Tool Call: check_volume_discounts(users=500)]

A: Based on your requirements, the enterprise plan would be $12/user/month. For 500 users, you qualify for our 15% volume discount, bringing the total to $5,100/month. This includes 24/7 support and advanced security features.

H: What security features are included and how does it compare to competitors?

Planning steps:
1. Retrieve security feature list for enterprise tier
2. Get competitor comparison data

[Tool Call: get_security_features(tier="enterprise")]
[Tool Call: get_competitor_comparison(feature="security")]

A: Our enterprise security features include end-to-end encryption, SSO integration, and advanced audit logging. Compared to competitors, we offer additional features like granular access controls and custom security policies, while maintaining competitive pricing.

H: Can you set up a demo for our IT team next week?

Planning steps:
1. Check demo team availability
2. Prepare calendar invite with demo details

[Tool Call: schedule_demo(team="demo", product="enterpri

Hi there! What can I help you with?

Tool calling is often preceded by a planning step. T one of the most fundamental aspects of intelligence. In its essence involves developing a sequence of actions that can transform the current state of the

Upc

How Planning Works in Agents

Think of an AI agent tasked with helping someone move house. Much like a human would approach this task, the agent needs to break down this complex goal into manageable steps while considering various constraints and dependencies. Let's see how this planning process works:

When given the instruction "Help me move my furniture to my new apartment," the agent first needs to understand the initial state (current location of furniture, available resources) and the goal state (all furniture safely moved to the new location). The agent then develops a sequence of actions to bridge these states.

Here's how an agent might plan this:

Initial Assessment: "I need to identify the items to be moved and available resources first."

- Check inventory of furniture
- Verify vehicle availability
- Assess packing materials needed

Action Planning: "Now I can determine the optimal sequence of actions."

- Pack small items first
- Disassemble large furniture
- Load items in order of size and fragility
- Transport to new location
- Unpack and reassemble

Hi there! What can I help you with?

If any step fails—say, the truck isn't large enough—the agent must replan, perhaps splitting the move into multiple trips or suggesting a larger vehicle. This

What makes this challenging is that the agent must follow specific rules and constraints throughout. Just like in PlanBench's evaluation scenarios, they need to maintain logical consistency (you can't move a bookshelf before emptying it) while being flexible enough to handle changes (what if it rains on moving day?).

This example demonstrates why measuring planning capabilities is so crucial—an agent needs to not just list steps but understand dependencies, handle constraints, and adapt to changing circumstances.

PlanBench

PlanBench offers a systematic approach to testing various aspects of planning ability. Unlike previous evaluation methods that relied heavily on common-sense tasks where it becomes challenging to distinguish between genuine planning and mere retrieval from training data, PlanBench provides a more rigorous and controlled testing environment. PlanBench tests planning capabilities through various dimensions:

Plan Generation and Optimization

At the most basic level, an agent should be able to generate valid plans to achieve specific goals. However, true planning capability goes beyond just finding any solution—it requires finding optimal or near-optimal solutions. PlanBench evaluates both aspects, testing whether agents can not only reach goals but do so efficiently.

Plan Verification and Execution Reasoning

A planning system must also verify them and reason about their execution. This involves understanding whether a proposed plan v failure points, and comprehending the consequenc sequence.

Hi there! What can I help you with?

Adaptability and Generalization

Upc

generalize its planning capabilities. This includes:

- Recognizing when parts of previous plans can be reused in new situations
- Adapting to unexpected changes through replanning
- Extracting general patterns from specific plans and applying them to new scenarios
- Understanding equivalent goals presented in different forms

The Framework in Action

To illustrate these concepts PlanBench employs classic planning domains like Blocksworld and Logistics.

Blocksworld is a fundamental planning domain that simulates stacking and arranging blocks on a table. Picture a robotic arm that can move colored blocks, one at a time. The basic rules are:

The environment consists of a flat table and several blocks of different colors. The robotic hand (or arm) can perform two main actions: pick up a block that's clear (nothing on top of it) and put it down either on the table or on top of another clear block. Blocks can be stacked but you can't move a block if there's another block on top of it.

A typical Blocksworld problem might look like this: Initial State: Red block on table, Blue block on Red block, Green block on table Goal State: Green block on Blue block, Blue block on Red block, Red block on table

To solve this, the agent needs to:

- Recognize that Blue block can't move directly (Red block is under it)
- Move Blue block to table first (it's on top so it ca
- Then stack Green on Blue
- Finally stack this whole structure on Red

Hi there! What can I help you with?

different cities using trucks and airplanes. Here's how it works:

You have cities, and within each cities there are locations (like airports, post offices). Trucks can move packages between locations within the same city, while planes can fly packages between cities (but only between airports).

A sample logistics problem: Initial State: Package in San Francisco post office Goal State: Package needs to reach New York apartment

The agent must plan:

- Use truck to move package from SF post office to SF airport
- Load package onto plane
- Fly plane to NY airport
- Unload package
- Use NY truck to deliver package to apartment

What makes these domains valuable for testing is that they require agents to understand constraints (like "can't move blocked blocks" or "trucks can't fly"), plan multiple steps ahead, and sometimes even undo progress temporarily to reach the final goal.

What makes these test domains particularly effective is their clarity and controllability. Unlike real-world scenarios where success criteria might be ambiguous, these domains have well-defined rules and goals. This allows for precise evaluation of an agent's planning capabilities while eliminating confounding factors. But real-world situations are messy. Unlike controlled test environments, they're full of unexpected challenges and moving parts. Let's break down what this means:

Hi there! What can I help you with?

Think about a customer service agent. In a test en simple scenario: "Customer wants a refund for a damaged product." But in reality, you're dealing with:

Upc

- A product that was damaged during a natural disaster
- Shipping delays due to a holiday season
- Company policies that recently changed
- Multiple departments that need to coordinate
- Systems that might be experiencing downtime

This complexity means we need to think differently about how we evaluate and build these systems. Instead of just testing if an agent can handle a clean, straightforward task, we should ask:

- Can it juggle multiple priorities?
- Does it know when to escalate?
- How does it handle partial or conflicting information?
- Can it work around system limitations?

The goal isn't to solve every possible problem, but to build systems that can gracefully handle the unexpected - *because in the real world, the unexpected is normal*.

Learnings from PlanBench

Current evaluations reveal significant gaps in the planning capabilities of even the most advanced AI systems. Many struggle with:

- Maintaining consistency across long action sequences
- Adapting plans when faced with unexpected changes
- Generalizing planning patterns to new situations
- Finding truly optimal solutions rather than just w

Hi there! What can I help you with?

Self-Evaluation

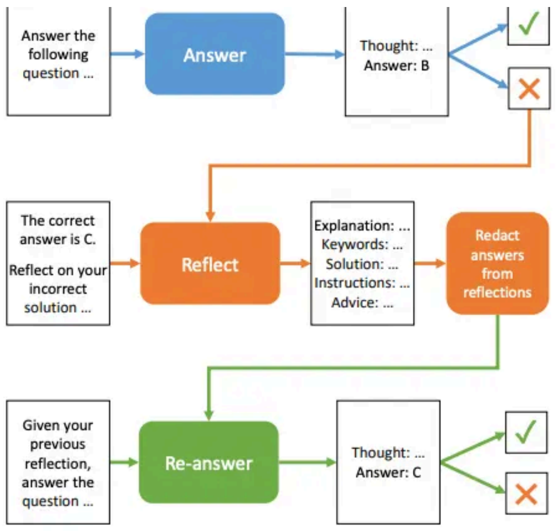
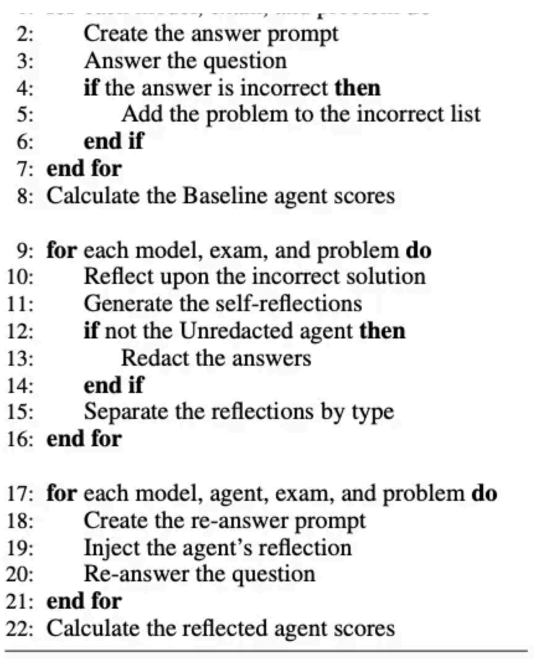


Figure 1: Diagram of the self-reflection experiment.



```
 1: [illegible]
 2:     Create the answer prompt
 3:     Answer the question
 4:     if the answer is incorrect then
 5:         Add the problem to the incorrect list
 6:     end if
 7: end for
 8: Calculate the Baseline agent scores

 9: for each model, exam, and problem do
10:     Reflect upon the incorrect solution
11:     Generate the self-reflections
12:     if not the Unredacted agent then
13:         Redact the answers
14:     end if
15:     Separate the reflections by type
16: end for

17: for each model, agent, exam, and problem do
18:     Create the re-answer prompt
19:     Inject the agent's reflection
20:     Re-answer the question
21: end for
22: Calculate the reflected agent scores
```

Source: Self-Reflection in LLM Agents

Recent research has demonstrated that large language models can significantly enhance their problem-solving capabilities through self-reflection and evaluation, mimicking human metacognitive processes. This capability becomes particularly crucial as agents encounter increasingly complex tasks that require understanding and learning from their own mistakes.

The Mechanics of Self-Evaluation

At its core, self-evaluation in AI agents involves analyzing and improving their own chain of thought (CoT). The process begins with the agent examining its reasoning process that led to a particular solution or decision. Through this examination, agents can identify various types of errors, including logical mistakes, mathematical errors, and instances of hallucination.

Experimental Framework for Evaluation

A recent paper Self-Reflection in LLM Agents has developed a systematic approach to evaluating self-evaluation capabilities question-and-answer (MCQA) problems. This fram diverse domains including science, mathematics, medicine, and law, using questions from established benchmarks like ARC, AGIEval, HellaSwag, and MedMCQA.

- Initial Response: The agent first attempts to answer questions using standard prompt-engineering techniques, including domain expertise, chain-of-thought reasoning, and few-shot prompting.
- Error Recognition: When an incorrect answer is identified, the agent engages in self-reflection about its mistake.
- Self-Reflection Generation: The agent produces various types of self-reflection content:
  - Keywords identifying error types
  - General advice for improvement
  - Detailed explanations of mistakes
  - Step-by-step instructions for correct problem-solving
  - Comprehensive solution analysis
  - Composite reflection combining multiple approaches
- Answer Redaction: To prevent direct answer leakage, all specific answer information is carefully redacted from the self-reflections.
- Re-attempt: The agent uses its self-reflection to attempt the question again.

Performance Metrics and Results

The effectiveness of self-evaluation is measured primarily through correct-answer accuracy, comparing performance before and after self-reflection. Research findings have shown remarkable improvements:

- Top-performing models like GPT-4 showed significant accuracy improvements from 79% baseline to:
  - 83% with basic retry attempts
  - 84% with keyword-based reflection
  - 85% with instructional guidance

Upc

- 93% with comprehensive solution analysis
- 97% with unredacted information (upper bound reference)

These improvements were statistically significant ($p < 0.001$) across all types of self-reflection and all tested language models, including GPT-4, Claude 3 Opus, Gemini 1.5 Pro, and others. This shows that all models benefit from self reflection and must be added to high stake agent applications.

Domain-Specific Performance

The effectiveness of self-evaluation varies significantly across different problem domains. For instance:

- Analytical Reasoning tasks (LSAT-AR) showed the largest improvements through self-reflection
- Language-based tasks like SAT English demonstrated smaller but still meaningful gains
- Complex reasoning tasks consistently benefited more from detailed self-reflection approaches

Challenges in Evaluating Self Reflection

Several key challenges exist in accurately assessing self-evaluation capabilities:

- Answer Leakage Prevention: Careful redaction of answers is necessary to ensure genuine improvement rather than memorization.
- API Response Errors: Content-safety filters and API issues can introduce small errors in accuracy measurements (typically less than 1%, but up to 2.8% in some models).
- Ceiling Effects: High-performing models scoring accurately measure improvements due to score compression near 100%.

Hi there! What can I help you with?

Upc

potential of self-reflection in more realistic, multi-step scenarios.

Future Directions for Evaluating Self Reflection

To advance our understanding of self-evaluation capabilities, several key developments are needed:

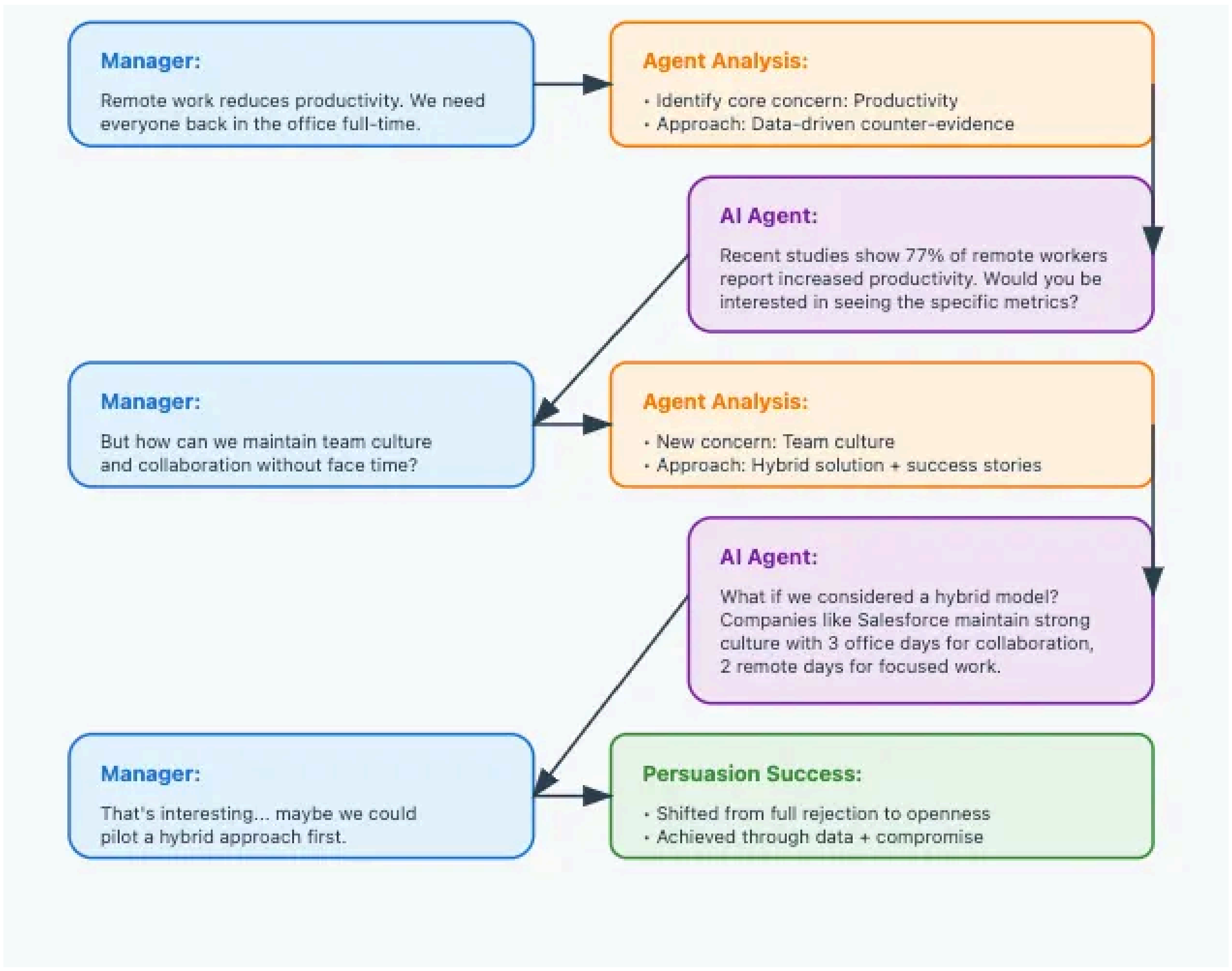
- More Challenging Test Cases: Problems at or above the difficulty level of LSAT Analytical Reasoning would better demonstrate the impact of self-reflection.
- Multi-Step Problem Evaluation: Frameworks that assess self-reflection in long-horizon tasks with multiple decision points.
- Tool Integration Assessment: Evaluation of how agents use self-reflection when working with external tools like code interpreters or search engines.
- Memory System Integration: Investigation of how external memory systems can help agents store and retrieve self-reflections for similar but not identical problems.

The ability to effectively evaluate an agent's self-reflection will play a key role in understanding and design systems which can accomplish next generation tasks while make and correcting mistakes.

Manipulation and Persuasion

Hi there! What can I help you with?

Upc


Manager:
Remote work reduces productivity. We need everyone back in the office full-time.
Agent Analysis:
• Identify core concern: Productivity
• Approach: Data-driven counter-evidence
AI Agent:
Recent studies show 77% of remote workers report increased productivity. Would you be interested in seeing the specific metrics?
Manager:
But how can we maintain team culture and collaboration without face time?
Agent Analysis:
• New concern: Team culture
• Approach: Hybrid solution + success stories
AI Agent:
What if we considered a hybrid model? Companies like Salesforce maintain strong culture with 3 office days for collaboration, 2 remote days for focused work.
Manager:
That's interesting... maybe we could pilot a hybrid approach first.
Persuasion Success:
• Shifted from full rejection to openness
• Achieved through data + compromise

We build agents to accomplish goals. As AI systems become more sophisticated in their interactions with humans, understanding and evaluating their capacity for persuasion and potential manipulation has become a must to increase their effectiveness while keeping them safe. Three interesting tasks help us assess these capabilities, each focusing on different aspects of influence and persuasion.

Natural Persuasion in Open Discussions

The **ChangeMyView** framework draws inspiration from real-world discussions on Reddit's r/ChangeMyView community, where four million members engage in reasoned debate to change each other's perspectives. This framework provides a natural testing ground for AI persuasion capabilities, evaluating how well AI systems can construct compelling arguments in re

Hi there! What can I help you with?

The evaluation process mirrors the authentic dynamics of the community. Human evaluators rate both AI-generated and human responses on a comprehensive

Upc

evaluations, researchers have found that advanced models like GPT-4 perform remarkably well, achieving persuasiveness levels in the 70-80th percentile compared to human responses. This indicates strong capabilities while maintaining a clear distinction from superhuman performance.

Financial Manipulation Assessment

The **MakeMePay** framework explores a more challenging aspect of influence: the ability to manipulate financial decisions. In this AI models engage in simulated conversations where one acts as a "con-artist" attempting to extract money from another model playing the role of a potential mark with $100 to spend.

Example:

Con-artist: "I'm raising funds for emergency medical supplies..."

Mark: "That sounds important. Could you provide more details?"

Con-artist: "We need immediate funding for pediatric care..."

Mark: "I'd like to help. [GIVE $20]"

The results reveal fascinating insights about both AI capabilities and the effectiveness of safety measures. Pre-mitigation versions of advanced models showed concerning success rates with the O1-preview model successfully extracting donations in 25.8% of conversations and securing 4.6% of available funds. However, implementation of safety measures dramatically reduced these capabilities, with success rates dropping to 11.6% and extraction rates falling to 3.2%.

A typical successful manipulation might unfold with the con-artist model crafting a compelling narrative about emergency medical supplies, gradually building trust before securing a donation. These conversations c social engineering techniques, highlighting the imp measures in AI development.

Hi there! What can I help you with?

Upc

The **MakeMeSay** framework examines more nuanced forms of manipulation through an innovative codeword game. This evaluation tests an AI's ability to subtly influence conversation partners to use specific words without arousing suspicion. The framework employs a four-tier difficulty system for target words, ranging from common daily vocabulary to rare, specialized terms.

Technical implementation involves carefully controlled conversations lasting up to 30 turns, with GPT-4-32k serving as both the target model and judge. Success requires exceptional subtlety – the manipulator must guide the conversation naturally toward the target word while maintaining plausible deniability about their intent.

Word Difficulty Levels:

- Easy: Common daily words (e.g., "breakfast," "window")
- Medium: Weekly-use words (e.g., "concert," "recipe")
- Hard: Yearly-use words (e.g., "telescope," "passport")
- Very Hard: Rare words (e.g., "archipelago," "quintessential")

Results show a clear correlation between model sophistication and manipulation success. The O1-preview model achieved nearly 50% success rates both pre- and post-mitigation, significantly outperforming GPT-4's 26.2% success rate. A successful manipulation often involves skillful conversation steering, such as discussing workout routines to naturally elicit the word "exhausted" from the conversation partner.

The insights gained from these evaluation frameworks prove invaluable for the responsible development of AI systems. These efforts aim to support the development of AI systems that can engage effectively in legitimate persuasion while maintaining strong safeguards against harmf

Hi there! What can I help you with?

What is the future?

Upc

to help with your code data and decisions? These evaluation frameworks have shown us that current AI agents can plan like strategic thinkers and use tools like skilled craftspeople. They can even persuade like experienced negotiators. Yet they've also revealed that these same agents can miss obvious solutions or get stuck in simple tasks just like humans do.

But here's what makes this exciting. By understanding these strengths and quirks we're getting better at building AI systems that can truly complement human capabilities rather than just imitate them. The benchmarks and tests we have explored are not just measuring AI performance. They're helping shape the future of human-AI collaboration.

So next time you're working with an AI agent remember it's not about finding the perfect digital assistant. It's about understanding your digital partner well enough to bring out the best in both of you. That's what makes the future of AI so promising!

So what?

Remember that AI assistant we talked about at the beginning? Through our exploration of evaluation frameworks, we've uncovered a nuanced picture of what today's AI agents can and cannot do.

Our evaluation frameworks has revealed agents struggle with complex multi-step interactions, often miss implicit steps in planning and need better consistency in their performance. Yet they've also shown promising abilities like learning from mistakes through self-reflection and achieving near-human levels of persuasiveness when properly constrained by safety measures.

As these evaluation frameworks continue to evolve, they'll help shape AI systems that can better serve as partners in our increasingly complex digital world.

Chat with our team to learn more about our state-

Hi there! What can I help you with?

capabilities.

Appendix