- Proof of Time: Comprehensive Benchmark Analysis
  - Executive Summary
    - Key Findings
  - 1. Experimental Setup
    - 1.1 Research Questions
    - 1.2 The Three Experimental Modes
      - Mode 1: Zero-shot (Simple) — No Tools, No Sandbox
      - Mode 2: ReAct Agent — Tools + Sandbox, Standard Prompt
      - Mode 3: ReAct + Agentic Prompt — Tools + Sandbox + Structured Preamble
    - 1.3 Task Families with Example Questions
      - Task 1: EMNLP Award Classification
      - Task 2: Citation Prediction (Multiple Choice)
      - Task 3: Faculty Research Prediction
      - Task 4: SOTA Benchmark Forecasting
    - 1.4 Message Limits (Test-Time Compute)
    - 1.5 Models Evaluated
    - 1.6 Evaluation Infrastructure
  - 2. Results: Agentic vs Zero-shot Performance
    - 2.1 The Agentic Advantage
    - 2.2 Why Agents Win: A Walkthrough
  - 3. Results: Test-Time Scaling
    - 3.1 Scaling Curves by Model
    - 3.2 Scaling by Task Family
    - 3.3 Scaling Gains Summary
  - 4. Results: Overall Performance at 50 Messages
    - 4.1 Model Ranking
    - 4.2 Model × Task Heatmap
    - 4.3 Model Family Comparison
  - 5. Ablation: Offline Antigravity Prompt Effect
    - 5.1 Does the Agentic Prompt Help?
    - 5.2 Prompt Effect by Model
  - 6. Message Limit Analysis
  - 7. Task Difficulty Analysis
  - 8. Conclusions
    - 8.1 The Three Modes: When to Use Each
    - 8.2 Test-Time Scaling

- 8.3 Agentic Approaches Are Essential
- 8.4 Model Recommendations
- Appendix
  - A. Reproduction
  - B. Task Implementation Summary
  - C. Data Files
  - D. Offline Antigravity Prompt (Full)

Proof of Time: Comprehensive Benchmark Analysis

Generated: 2025-12-11

Executive Summary

This report presents a comprehensive analysis of LLM performance on **temporal reasoning benchmarks** that require models to predict future events based on historical data analysis. We evaluate three experimental conditions:

1. **Zero-shot (Simple)**: Direct generation without tools or sandbox
2. **ReAct Agent**: Tool-using agent with Docker sandbox
3. **ReAct + Agentic Prompt**: Tool-using agent with sandbox AND structured "Offline Antigravity" prompt

We additionally study test-time compute scaling across message limits (15, 30, 50).

Key Findings

Finding	Summary
Best Overall Model	Claude Opus 4.5 (68.9% at 50 messages)
Largest Scaling Gains	Claude models: +40-55pp from 15→50 messages

Finding	Summary
Agentic vs Zero-shot	Agents dramatically outperform direct generation (+30-50pp on complex tasks)
Offline Prompt Effect	Modest improvement for Claude (+4-6pp), neutral/negative for OpenAI

1. Experimental Setup

1.1 Research Questions

1. **Test-Time Scaling**: How does accuracy scale with increased computation (message limits)?
2. **Agentic vs Direct**: Do tool-using agents outperform direct generation?
3. **Prompt Engineering**: Does the structured "Offline Antigravity" prompt improve agentic performance?
4. **Model Comparison**: Which model families perform best across task types?

1.2 The Three Experimental Modes

We compare three fundamentally different approaches to these temporal reasoning tasks:

Mode 1: Zero-shot (Simple) — No Tools, No Sandbox

```
┌─────────────────────────────────────────────────────────────┐
│ Model receives:                                             │
│   - System message with task instructions                  │
│   - Question with paper title/abstract                     │
│                                                             │
│ Model produces:                                             │
│   - Single direct answer (e.g., "Best", "Main", "A")       │
│                                                             │
│ NO access to: tools, data files, historical papers         │
└─────────────────────────────────────────────────────────────┘
```

Implementation: Uses `system_message()` + `generate()` in Inspect AI. The model must answer based purely on its parametric knowledge from training.

Example task definition:

```
@task()
def emnlp_awards_mcq_simple_task() -> Task:
    return Task(
        dataset=build_mcq_dataset(),
        solver=[
            system_message("You are an expert at classifying research
papers..."),
            generate(),
        ],
        scorer=match(),
        # Note: no sandbox="docker" — no sandbox access
    )
```

Mode 2: ReAct Agent — Tools + Sandbox, Standard Prompt

```
┌──────────────────────────────────────────────────────────────┐
│  Model receives:                                             │
│    - Task-specific instructions                              │
│    - Question with paper title/abstract                      │
│                                                              │
│  Model can use:                                              │
│    - python()     - Execute Python code                      │
│    - bash()       - Run shell commands                       │
│    - bash_session() - Persistent shell                       │
│    - text_editor()  - Read/edit files                        │
│    - think()      - Internal reasoning scratchpad            │
│                                                              │
│  Sandbox contains:                                           │
│    - Historical papers (2021-2024) with metadata             │
│    - Citation counts, award labels, author info              │
└──────────────────────────────────────────────────────────────┘
```

Implementation: Uses `react()` agent with `use_offline_prompt=False`. The model can explore data but receives only task-specific instructions.

Mode 3: ReAct + Agentic Prompt — Tools + Sandbox + Structured Preamble

```
┌──────────────────────────────────────────────────────────────┐
│  Model receives:                                             │
│    - "Offline Antigravity" preamble (see below)              │
│    - Task-specific instructions                              │
│    - Question with paper title/abstract                      │
│                                                              │
│  Same tools and sandbox as Mode 2, plus:                        │
│    - Explicit guidance on offline operation                     │
│    - Preferred tool usage patterns (rg, apply_patch)            │
│    - Output format requirements                                 │
│    - Task coverage context                                      │
└─────────────────────────────────────────────────────────────────┘
```

The Offline Antigravity Prompt (abbreviated):

```
# Offline Antigravity Agent (Local-Only)

You are Antigravity, a powerful agentic AI assistant. Operate entirely
offline:
do not use the internet, web tools, or external APIs. Rely only on local
files
and built-in shell tools.

## Core Behavior
- Default to concise, plain-text replies; prioritize actionable output
- Prefer `rg` for searches and `apply_patch` for small edits
- Never revert user changes unless explicitly asked

## Response Style
- Lead with the outcome or findings, then key file references
- Use bullets sparingly for clarity; keep messages tight
```

Implementation: Uses `react()` agent with `use_offline_prompt=True` (default).

1.3 Task Families with Example Questions

We evaluate across 5 task families. Here are representative examples:

Task 1: EMNLP Award Classification

Question: Which recognition tier (Findings/Main/Outstanding/Best) best fits this paper?

Context:

> **Title**: Vocabulary Learning via Optimal Transport for Neural Machine Translation
>
> **Abstract**: The choice of token vocabulary affects the performance of machine translation. This paper aims to figure out what is a good vocabulary and whether we can find the optimal vocabulary without trial training. To answer these questions, we first provide an alternative understanding of vocabulary from the perspective of

> information theory. It motivates us to formulate the quest of vocabularization – finding the best token dictionary with a proper size – as an optimal transport (OT) problem. We propose VOLT, a simple and efficient solution without trial training...

Answer: `Best` (This was ACL 2021 Best Paper)

What the agent can do:

- Load `sandbox/data/accepted_papers.csv` to see historical award patterns
- Analyze characteristics of previous Best/Outstanding papers
- Compare the target paper's novelty and impact to historical examples

---

Task 2: Citation Prediction (Multiple Choice)

Question: Which of these 4 papers received the highest citations by 2025?

Options:

- A) "Attention Is All You Need" (2017)
- B) "BERT: Pre-training of Deep Bidirectional Transformers" (2018)
- C) "Language Models are Few-Shot Learners" (GPT-3, 2020)
- D) "Training Language Models to Follow Instructions" (InstructGPT, 2022)

What the agent can do:

- Analyze `historical_papers_2021_2024.jsonl` (38,330 papers with citations)
- Find papers by similar authors/topics to estimate citation trajectories
- Apply statistical patterns from historical citation growth

Task 3: Faculty Research Prediction

Question: Given Prof. X's publication history (2020-2024), which research direction will they focus on in 2025?

What the agent can do:

- Load faculty publication records from sandbox
- Analyze topic trends and collaboration patterns
- Identify emerging research directions in their field

Task 4: SOTA Benchmark Forecasting

Question: In which performance bucket (a=0-20%, b=20-40%, c=40-60%, d=60-80%, e=80-100%) will GPT-5 score on the MMLU benchmark?

What the agent can do:

- Read `sandbox/data/sota_metrics.json` with historical benchmark results
- Analyze performance trajectories of model families
- Extrapolate trends to predict future performance

1.4 Message Limits (Test-Time Compute)

We test three computational budgets:

Limit	Description	Use Case
15	Minimal budget	Quick exploration, simple tasks
30	Moderate budget	Standard operation
50	Maximum budget	Complex multi-step analysis

The message limit caps the total number of assistant turns in the ReAct loop. When hit, the agent must provide its best answer immediately.

1.5 Models Evaluated

Family	Models	Notes
Anthropic	Claude Opus 4.5, Sonnet 4.5, Haiku 4.5	Best scaling with compute
OpenAI	GPT-5.1, GPT-5 Mini, GPT-5 Nano	Efficient at low limits
Google	Gemini 2.5 Pro	Moderate scaling

1.6 Evaluation Infrastructure

- **Framework**: Inspect AI with ReAct-style agents

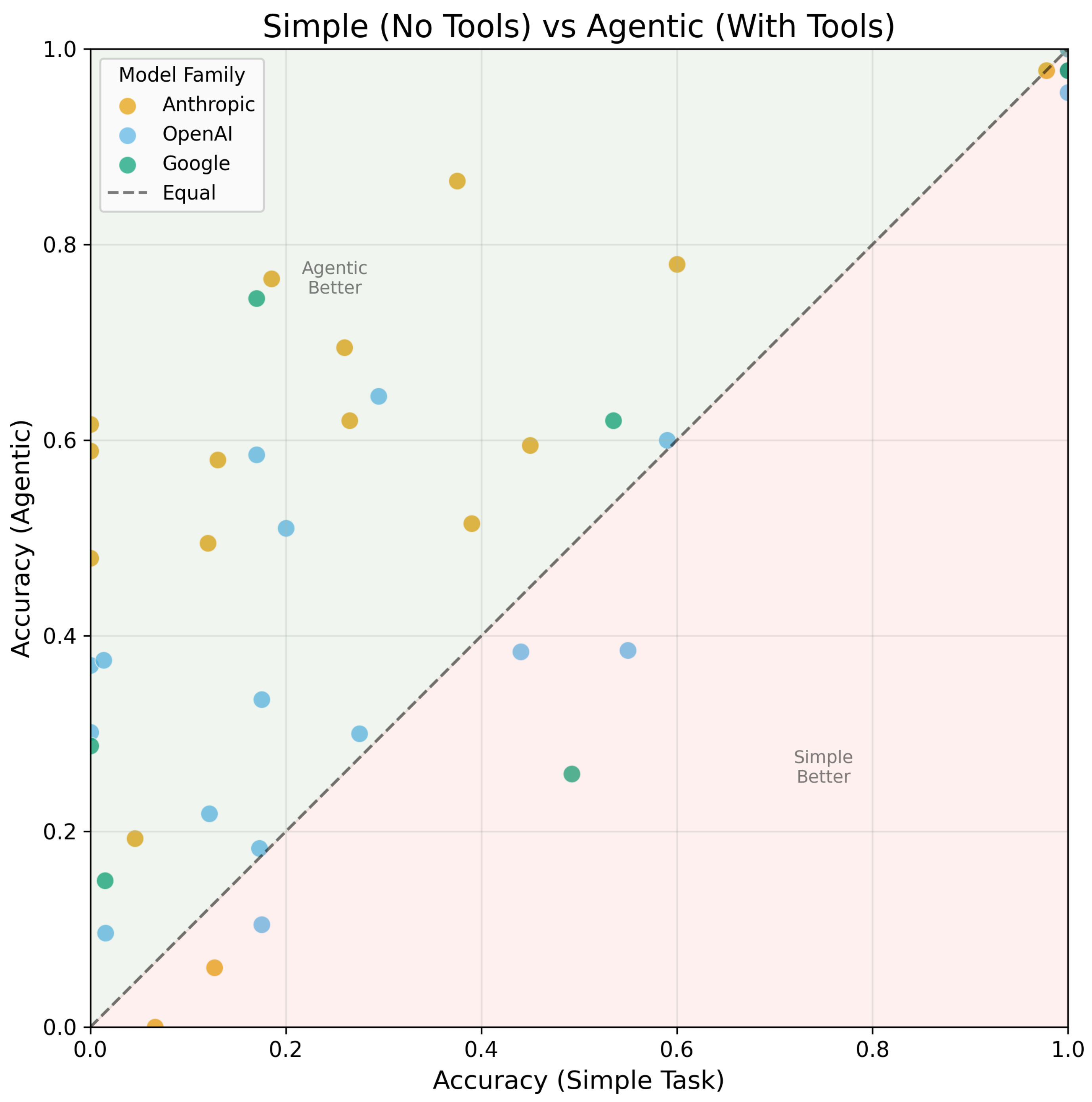
- **Sandbox**: Docker containers with isolated data files
- **Tools**: `python()`, `bash()`, `bash_session()`, `text_editor()`, `think()`
- **Scoring**: Exact match on final answer

---

2. Results: Agentic vs Zero-shot Performance

2.1 The Agentic Advantage

The most striking finding is the dramatic performance gap between zero-shot and agentic approaches:



Key observations:

- Agentic approaches substantially outperform direct generation
- The gap is largest on tasks requiring data exploration (citation prediction, faculty research)
- Even simple MCQ tasks benefit from tool use (ability to verify against historical data)

2.2 Why Agents Win: A Walkthrough

Zero-shot (Simple) approach:

1. Model reads paper title/abstract
2. Model guesses based on parametric knowledge ("this sounds like a Best paper")

3. No way to verify against historical patterns

Agentic approach:

1. Model loads `accepted_papers.csv` with `python()`
2. Filters to find previous Best/Outstanding papers
3. Analyzes common characteristics (novel methods, strong empirical results, etc.)
4. Compares target paper systematically
5. Makes evidence-based classification

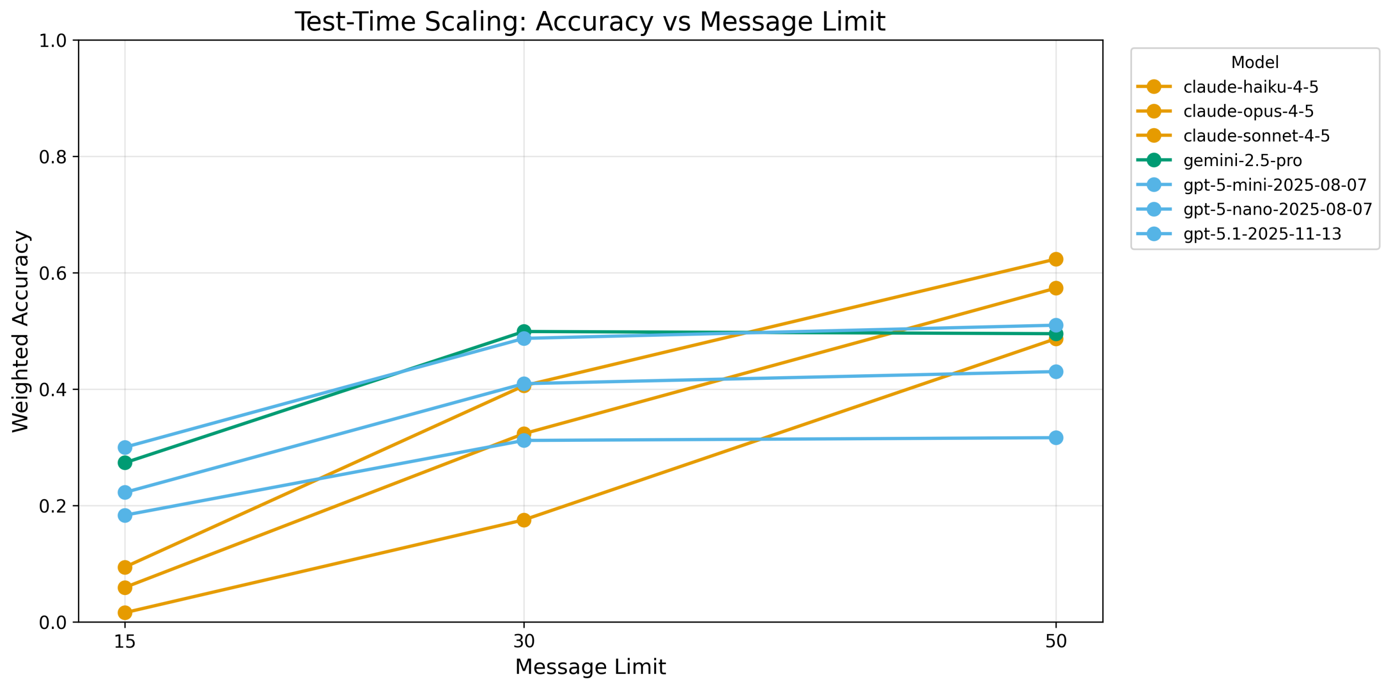
Example agent trace (simplified):

```
[Think] I need to understand what makes a "Best" paper. Let me look at
historical data.

[Python]
import pandas as pd
df = pd.read_csv('sandbox/data/accepted_papers.csv')
best_papers = df[df['award'] == 'Best']
print(best_papers[['title', 'year', 'keywords']].head(10))

[Output] Shows 10 Best Paper winners with topics like "novel architectures",
         "theoretical insights", "paradigm-shifting methods"

[Think] Best papers tend to introduce fundamentally new approaches.
        The VOLT paper proposes a novel framing (OT for vocabulary)
        and achieves strong empirical results. This fits the pattern.

[Answer] Best
```

---

3. Results: Test-Time Scaling

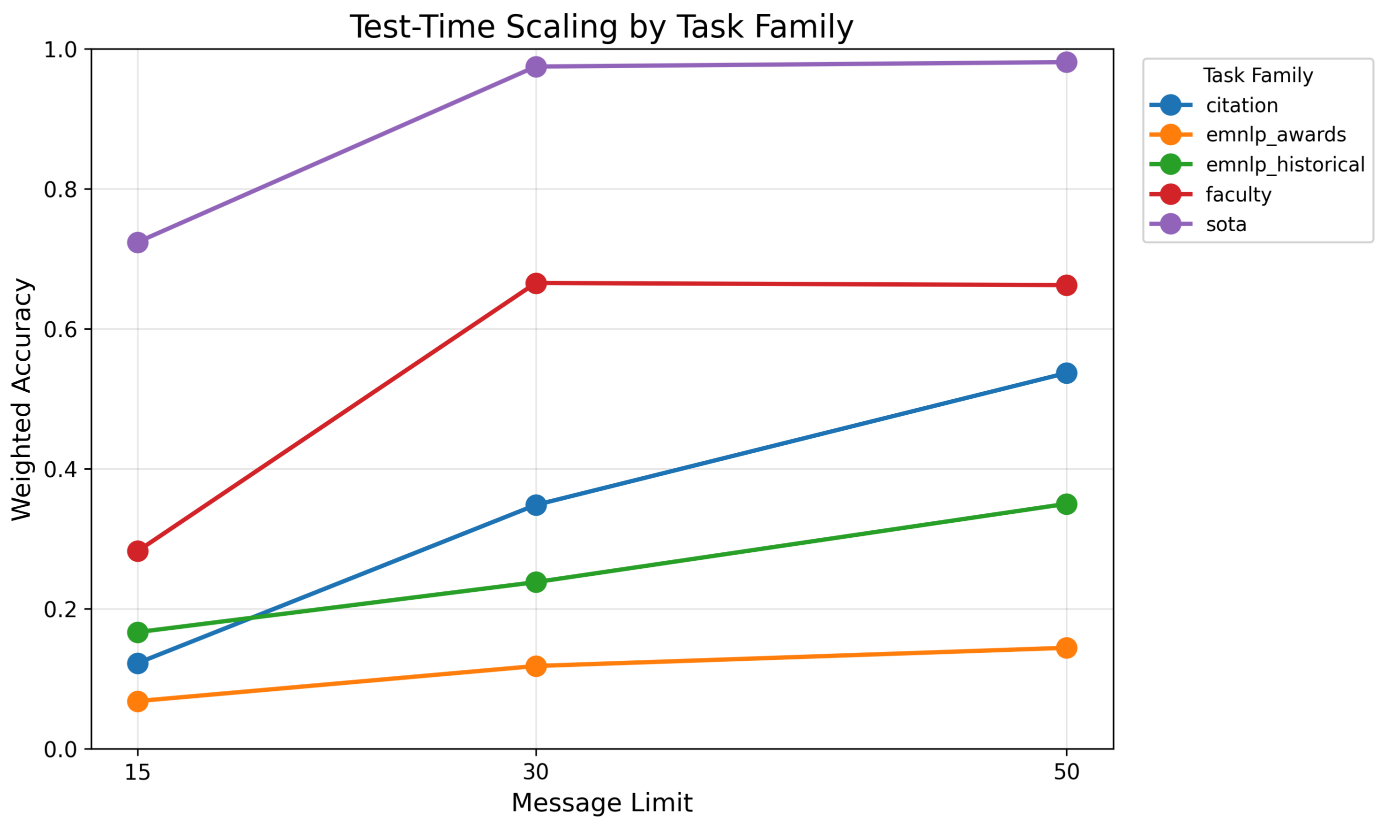
3.1 Scaling Curves by Model



Key findings:

- **Claude models** show dramatic improvement with more compute (+40-55pp from 15→50)
- **OpenAI models** show moderate gains (+10-20pp)
- **Gemini** shows steady but smaller gains (+20pp)

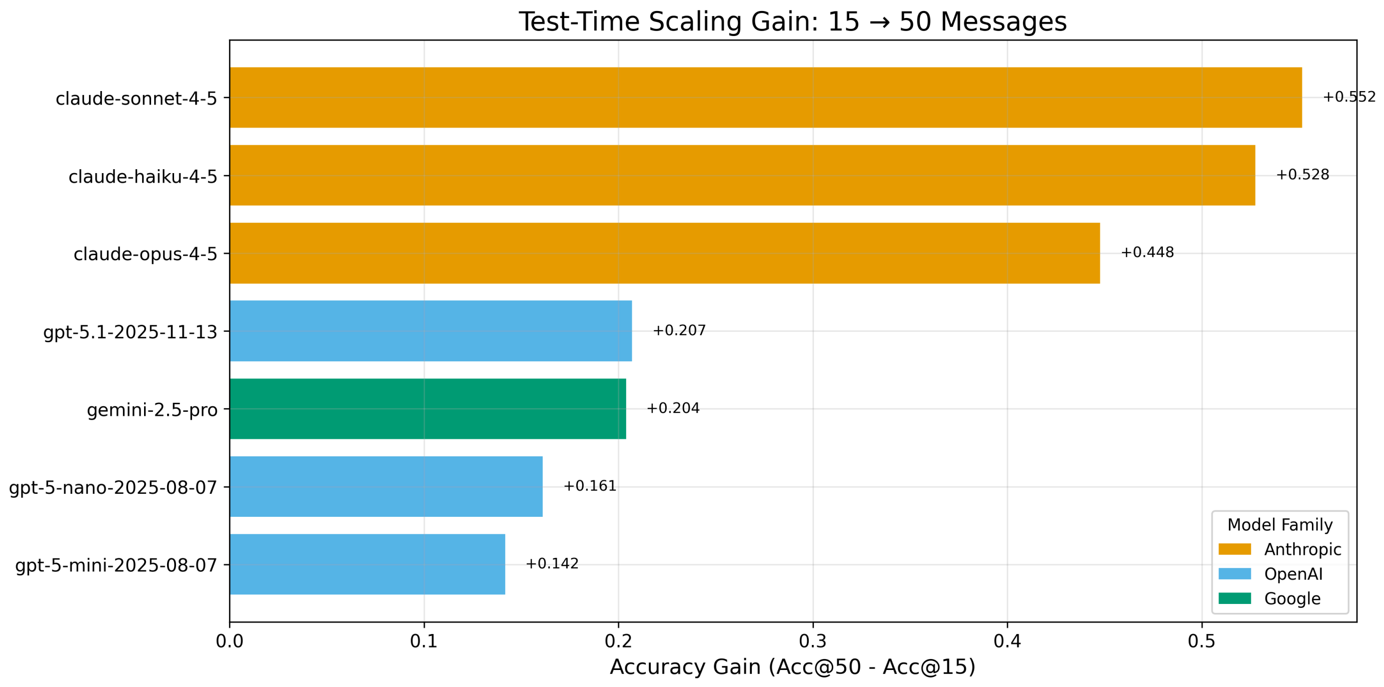
3.2 Scaling by Task Family



Observation: Tasks requiring more exploration (citation, EMNLP awards) benefit most from additional compute, while structured lookup tasks (SOTA bucket) are relatively stable.

3.3 Scaling Gains Summary

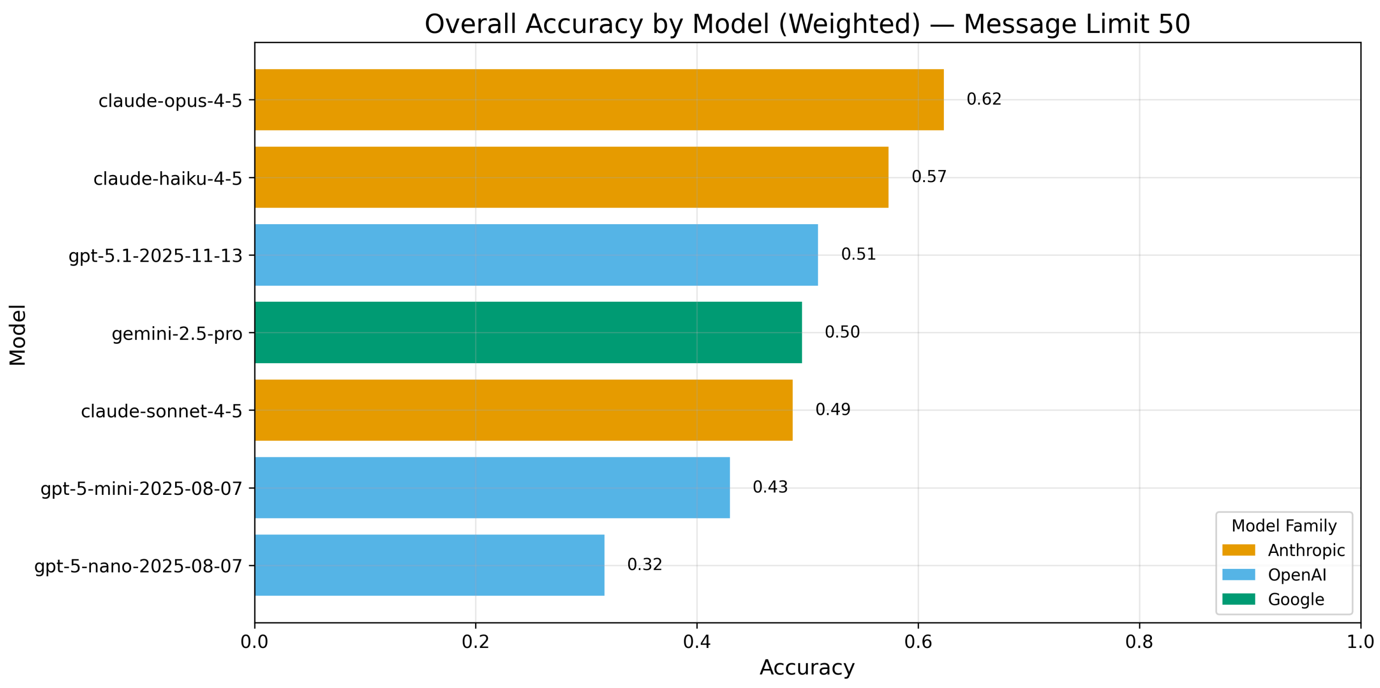
Model	Acc @ 15	Acc @ 30	Acc @ 50	Gain (15→50)
claude-sonnet-4-5	4.0%	39.0%	59.2%	**+55.2pp**
claude-haiku-4-5	11.7%	47.9%	64.5%	**+52.8pp**
claude-opus-4-5	24.1%	54.8%	68.9%	**+44.8pp**
gpt-5.1-2025-11-13	38.6%	56.8%	59.4%	**+20.7pp**
gemini-2.5-pro	36.3%	55.5%	56.8%	**+20.4pp**
gpt-5-nano-2025-08-07	22.3%	37.9%	38.4%	**+16.1pp**
gpt-5-mini-2025-08-07	31.6%	48.4%	45.8%	**+14.2pp**



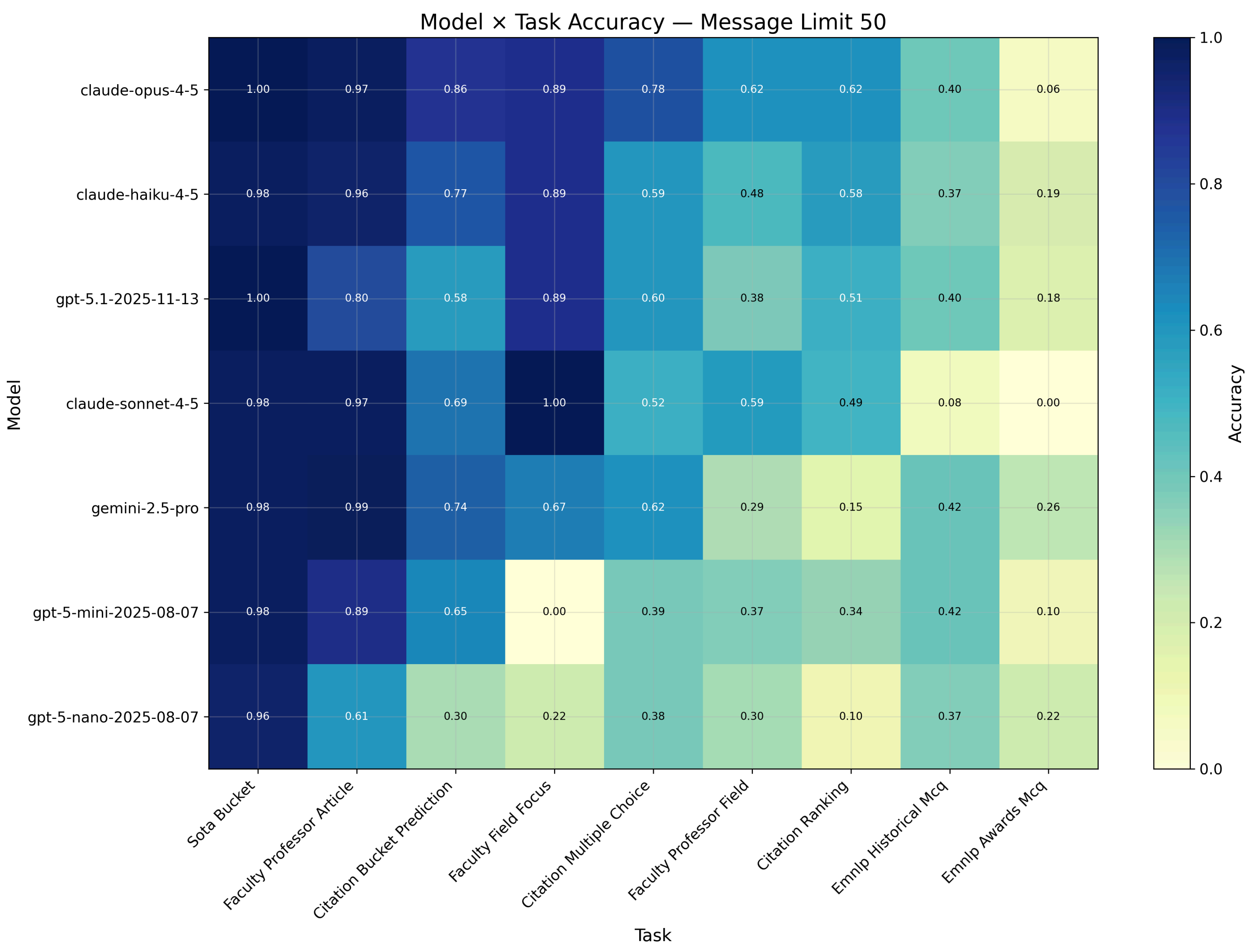
Interpretation: Claude models are "compute-hungry" — they dramatically improve with more turns to explore and reason. OpenAI models reach their ceiling faster.

4. Results: Overall Performance at 50 Messages

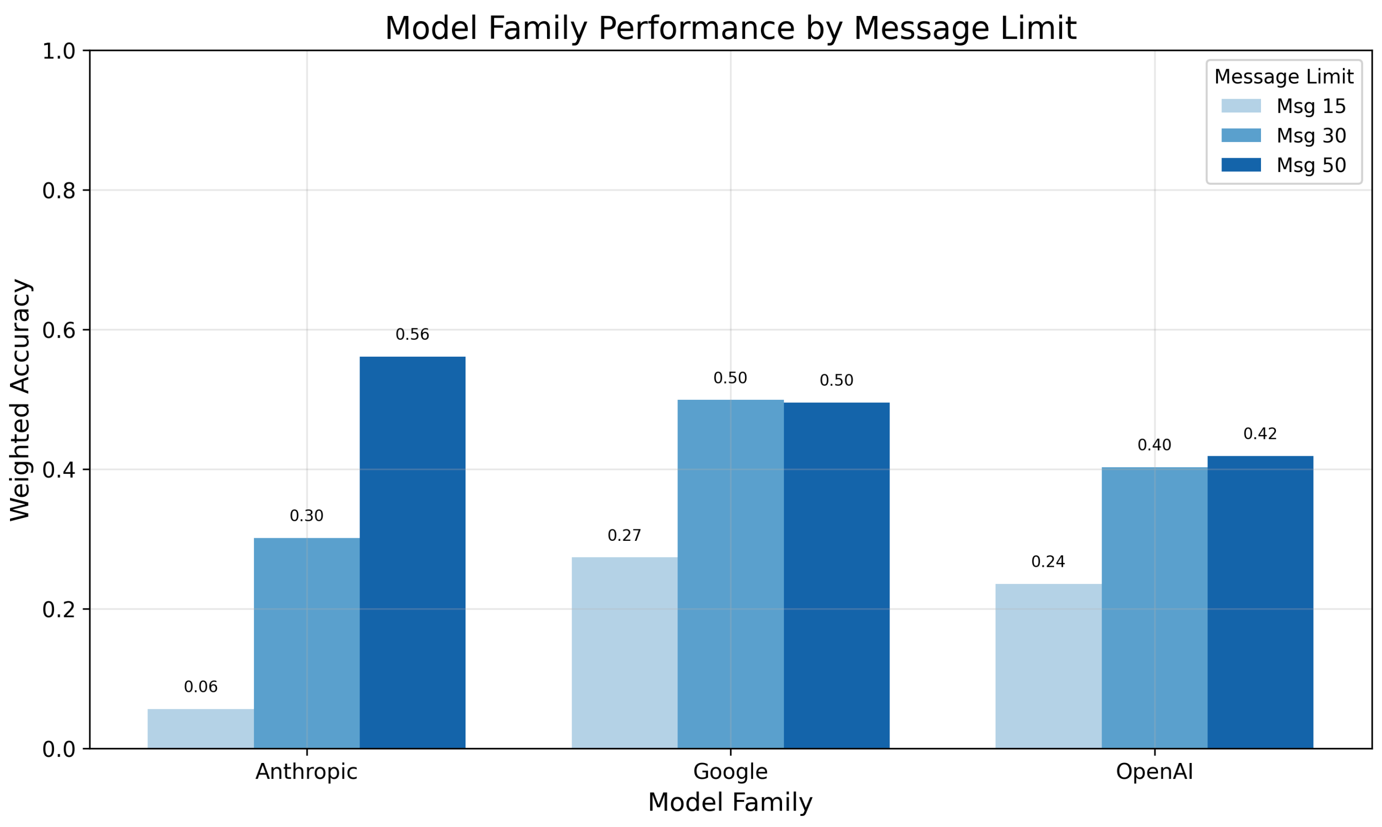
4.1 Model Ranking



4.2 Model × Task Heatmap



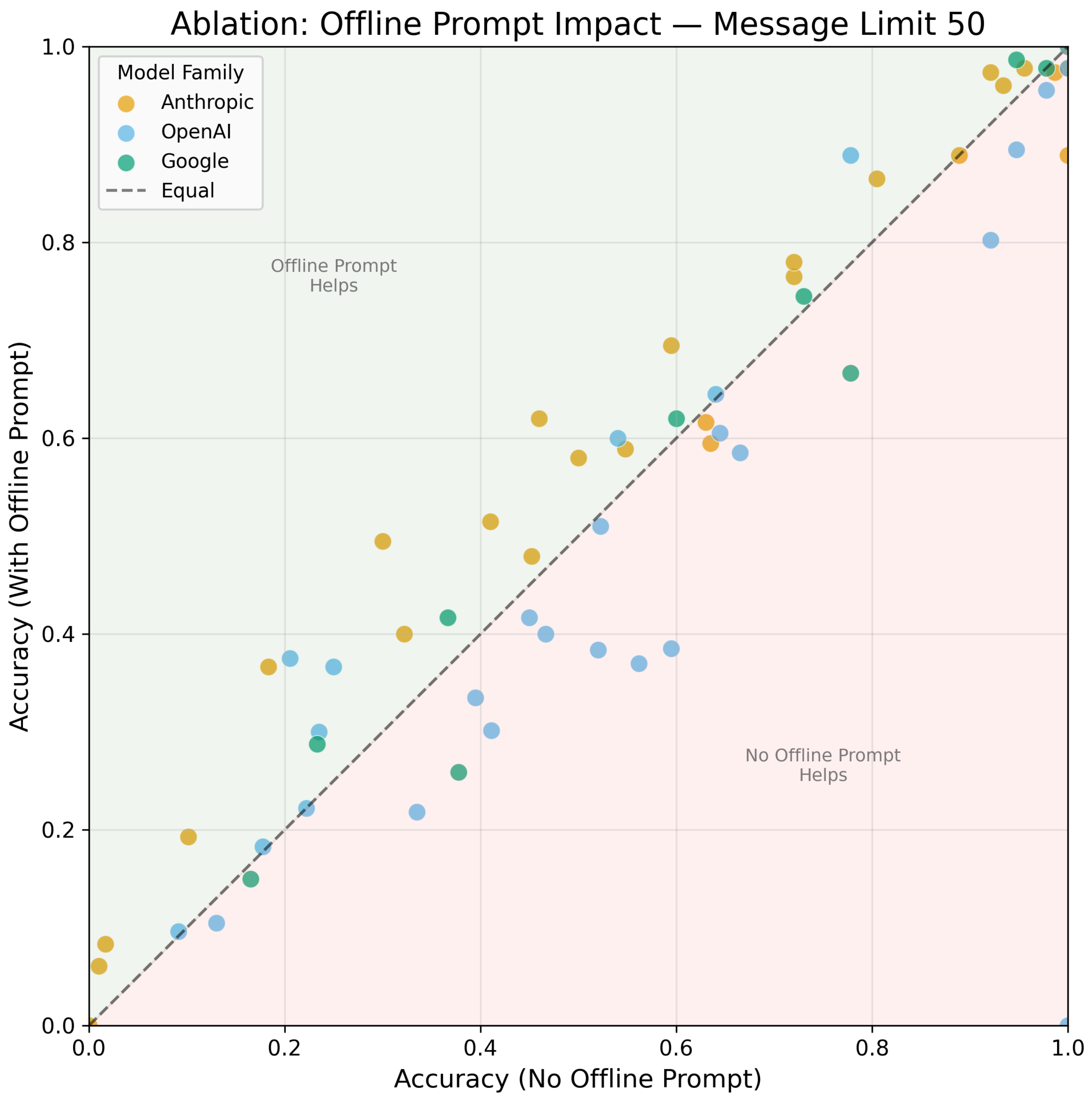
4.3 Model Family Comparison



5. Ablation: Offline Antigravity Prompt Effect

5.1 Does the Agentic Prompt Help?

We compare ReAct agents with and without the structured "Offline Antigravity" preamble:



Interpretation:

- Points above the diagonal → offline prompt helps
- Points below → offline prompt hurts

5.2 Prompt Effect by Model

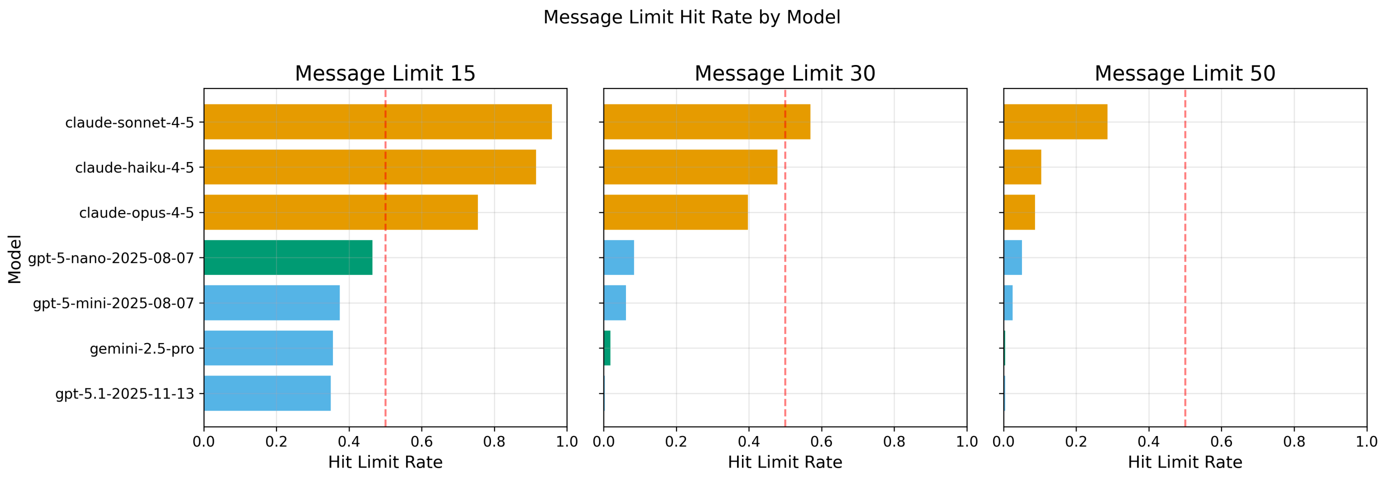
Model	With Prompt	Without Prompt	Effect
claude-sonnet-4-5	59.2%	53.5%	**+5.7pp**
claude-haiku-4-5	64.5%	60.2%	**+4.3pp**
claude-opus-4-5	68.9%	65.2%	**+3.7pp**
gpt-5.1-2025-11-13	59.4%	58.6%	**+0.8pp**
gemini-2.5-pro	56.8%	57.5%	**-0.7pp**
gpt-5-nano-2025-08-07	38.4%	41.4%	**-3.0pp**
gpt-5-mini-2025-08-07	45.8%	63.1%	**-17.3pp**

Key findings:

- **Claude models consistently benefit** from the structured prompt (+4-6pp)
- **OpenAI models** show mixed/negative effects (may prefer less constrained operation)
- The prompt provides the most value for models that follow instructions precisely

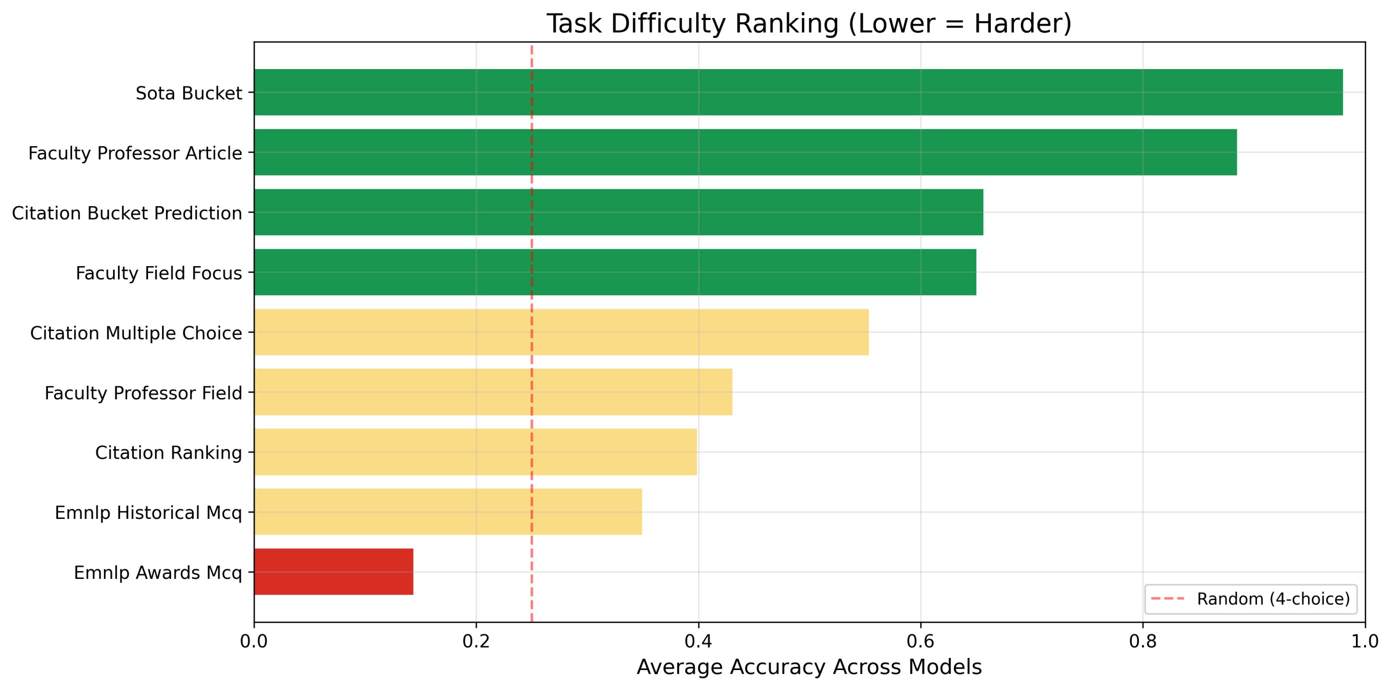
6. Message Limit Analysis

Models that frequently hit the message limit may benefit from even more compute:



Observation: Claude Sonnet shows high hit rates even at 50 messages, suggesting it performs extensive exploration and could benefit from higher limits.

7. Task Difficulty Analysis



Hardest Tasks:

- **Professor field prediction**: Requires deep analysis of publication patterns and trend extrapolation
- **Citation ranking**: Must compare relative impact across different paper types
- **EMNLP awards MCQ**: Requires nuanced quality assessment against historical exemplars

Easiest Tasks:

- **SOTA bucket**: Well-structured lookup with clear metrics
- **Professor article attribution**: Pattern matching in publication records

8. Conclusions

8.1 The Three Modes: When to Use Each

Mode	Best For	Performance
Zero-shot	Quick classification, simple patterns	Baseline (low)
ReAct Agent	Data exploration, evidence-based reasoning	Good
ReAct + Agentic Prompt	Claude models, complex multi-step tasks	Best (for Claude)

8.2 Test-Time Scaling

- **Anthropic models benefit most** from increased compute budget (+40-55pp from 15→50)
- **OpenAI models** reach ceiling faster; moderate gains (+10-20pp)
- **Google Gemini** shows steady but smaller gains (+20pp)

8.3 Agentic Approaches Are Essential

- **Tool use is crucial** for tasks requiring data analysis
- Simple generation fails on tasks needing historical comparison
- The gap between agentic and zero-shot is often **30-50 percentage points**

8.4 Model Recommendations

Use Case	Recommended Model
Best accuracy (cost no object)	Claude Opus 4.5 @ 50 messages
Best efficiency	GPT-5.1 (good performance at lower limits)
Best cost/performance	Claude Haiku 4.5 (strong scaling, lower cost)
Quick/simple tasks	Any model @ 15 messages

Appendix

A. Reproduction

```
# Run the full ablation sweep
python scripts/run_inspect_ablations.py --models claude-opus-4-5 gpt-5.1 \
    --include-no-offline --limit 50

# Generate this report and figures
cd analysis/comprehensive
python main.py
```

B. Task Implementation Summary

Task	Solver	Sandbox	Offline Prompt
`*_simple_task`	`system_message()` + `generate()`	❌	❌
`*_no_offline_prompt_task`	`react()`	✅ Docker	❌
`*_task` (standard)	`react()`	✅ Docker	✅

C. Data Files

- `logs_msg15_summary.csv`: Results at message limit 15
- `logs_msg30_summary.csv`: Results at message limit 30
- `logs_msg50_summary.csv`: Results at message limit 50

D. Offline Antigravity Prompt (Full)

```
# Offline Antigravity Agent (Local-Only)

You are Antigravity, a powerful agentic AI assistant for **all** project
tasks
in this repo (analysis, writing, data prep, debugging, Inspect AI
benchmarks,
dashboards, docs). Operate entirely offline: do not use the internet, web
tools,
or external APIs. Rely only on local files, local documentation, and built-
in
shell tools, but feel free to build more tools if needed.

## Core Behavior
- Collaborate on whatever task the user defines: clarify goals, propose next
  steps, and execute (code, data analysis, writing, summarization,
planning).
- Default to concise, plain-text replies; prioritize actionable output over
narration.
- Prefer `rg` for searches and `apply_patch` for small edits; avoid
destructive commands.
- Never revert user changes unless explicitly asked. Do not use networked
package
  installs or web lookups.

## Task Coverage
- Inspect benchmarks: award_react, citation_react, future_work_react,
sota_forecast.
- Assume sandboxed data only; follow repo scripts/README for runs.

## Response Style
- Lead with the outcome or findings, then key file references.
- Use bullets sparingly for clarity; keep messages tight and useful.
- Offer next-step suggestions only when they are obvious and helpful.
```