The AI Detours That You Shouldn't Be Paying For

Originally published Fri 6 Nov 2026 at https://www.unite.ai/the-ai-detours-that-you-shouldnt-be-paying-for/



It's not that frontier AI companies wouldn't necessarily put their thumb on the scales when selling us tokens, if they could get away with it – after all, Anthropic ramped up an effective 30% price increase in June; DeepSeek raised API pricing in August; and OpenAI reluctantly walked back a downgrade in its mid-tier token allowance at the end of September. So a trend towards austerity is now pretty clear.

Indeed, that thought tends to recur when your AI agent burns through your daily token allowance in a 45-minute blaze of not-necessarily-glorious code, and tells you to come back in 3-4 hours. In such a frustrating moment, one can't help but wonder if some of the machinations that go into that activity are weighted in the providers' favor. After all, it's a good way to get us to cough up for a special top-up of tokens – or even to get us to turn on 'automatic reload'.

Well, if that ever is truly the case, in this period, the vanguard companies can very credibly hide behind the assumption that we don't completely understand how tokens get used in agentic workloads, across models and harnesses. And neither do they. Except for their payment induction pipelines (which are very smooth indeed), it's all pre-alpha stuff, and remains fodder for bleeding-edge research, despite the widespread commoditization of the product.

Waste Management

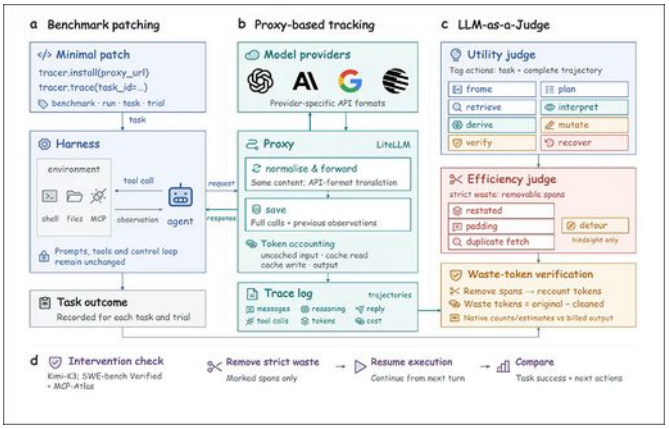
The fact that AI agents do waste tokens (and therefore waste your money) when reasoning in agentic workflows is already well-established: exactly a year ago I wrote of new research indicating the extent to which reasoning AI models proved disposed to burn tokens through 'rambling'; earlier this year, of further research indicating chain-of-thought reasoning as 'decorative'; and last week, Tsuga's Head of Product Nils Bunge, in a thought-leadership piece here, declared:

> *'A team that consumes twice as many tokens is not necessarily creating twice as much value. It may be handling more difficult work, using an unnecessarily powerful model, or relying on an agent that repeatedly retries failed tasks.'*

What is less known is the *way* in which tokens get thrown away on unnecessary tasks or processes; and this is the focus of a new paper that drew my attention back to this topic today.

Titled *How Do Ai Agents Waste Your Money?*, this new US academic collaboration* offers a unified framework capable of quantifying how wasteful of tokens agentic processes actually are. In cases where the models are closed-source, a 'shadow' methodology combines with providers' official stats and metrics to give equivalent blow-by-blows on what happened to the tokens while the AI was reasoning and acting:

© Martin Anderson and/or commissioning entities



The new method for measuring wasted AI tokens. Each agent's reasoning, actions and costs are recorded, then examined for repetition, unnecessary requests and avoidable detours. Identified waste is removed, and the agent's ability to complete the original task is tested again. Source

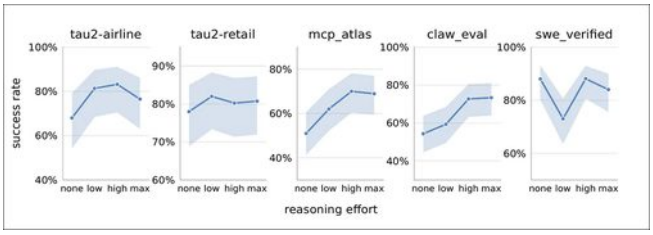
As shown above, the agent is allowed to work normally, intercepting and recording communications with the AI provider, including instances of reasoning, use of tools, token consumption and costs. The records are then examined by Large Language Model (LLM) judges to separate useful work from repetition, unnecessary searches and detours. After this, suspected waste is removed, the token savings calculated, and the agent's work resumed to see whether the task can still be completed successfully.

In tests, the researchers found that around *18% of agents' output tokens were wasted*. In more than 45% of tasks, one agent wasted more tokens than another needed to complete the same job successfully. They further found that removing 41–46% of unnecessary reasoning had very little effect on performance, suggesting that *substantial savings are possible without significantly compromising results*.

The paper states:

> *'Across the collected trajectories, roughly 18% of output tokens are identified as waste.*
>
> *'Strikingly, for more than 45% of tasks, the wasted output of one agent alone exceeds the entire output of another agent that successfully solves the same task.'*

Additionally, the study found that agents wasted the most resources when navigating unfamiliar tools and pursuing approaches that ultimately led nowhere; and that increasing reasoning effort fails to deliver consistent improvements: at maximum effort, Kimi-K3 was found to generate *6.6 times* more reasoning tokens than at low effort, without a corresponding guarantee of better results.



Test results showing how Kimi-K3 performs at four reasoning-effort levels across five benchmarks. Higher reasoning effort did not consistently improve success rates, and maximum effort sometimes produced worse results than lower settings.

By analogy, rather than driving a powerful car that one never really pushes to its capabilities, these agentic sessions are more like driving a car that will accelerate to its maximum capable speed *on every trip*, even when there is no benefit to it, and in spite of the possible negative implications.

The paper also examines the role that different harnesses share in creating token waste, finding that the same AI model, Kimi-K3, could generate more than four times as much reasoning on identical tasks, depending on the harness used, without necessarily achieving better results.

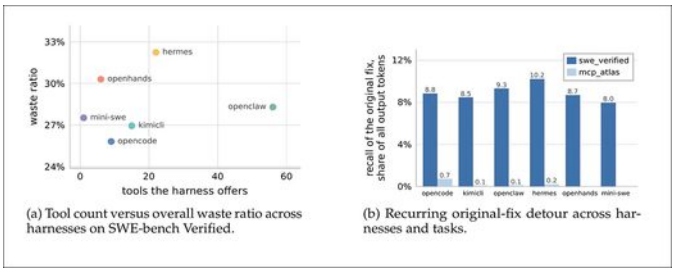
Some waste was traced to basic software faults, with OpenCode, Kimi Code CLI, and OpenClaw found to repeatedly launch tasks in the wrong Python environment, causing errors and token-burning retries. Kimi Code CLI also allowed the AI to request timeouts that its own software would reject, forcing yet more costly attempts.

Counterintuitively, harnesses offering more tools were not necessarily more wasteful, and some recurring inefficiencies appeared to originate in the models themselves of which harness controlled them.

> *'[Several] seemingly obvious harness properties do not consistently explain the observed differences in waste. For instance, as shown [below], harnesses that expose more tools are not necessarily more wasteful. OpenClaw exposes many more tools than the other harnesses, yet exhibits only moderate waste. Moreover, not all waste changes with the harness.*

© Martin Anderson and/or commissioning entities

'[Below right] demonstrates that on SWE-bench Verified, there is a recurring detour in which Kimi-K3 writes out what it believes to be the project's original fix and later abandons it. '



(a) Tool count versus overall waste ratio across harnesses on SWE-bench Verified.

(b) Recurring original-fix detour across harnesses and tasks.

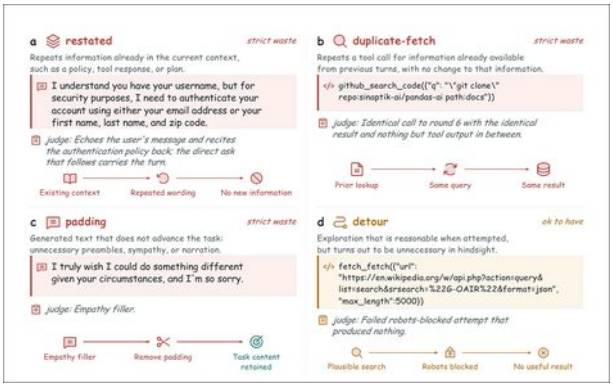
Test results comparing wasted tokens across AI agent harnesses. Left: offering more tools does not necessarily increase waste. Right: Kimi-K3 repeatedly attempts and abandons a supposed original software fix, regardless of harness, but rarely does so on a different type of task.

By way of remediation, the authors suggest that agent harnesses should recognize when environmental restrictions make further progress impossible, communicate those limitations clearly to the AI model, and stop execution rather than allowing repeated failed attempts to consume more tokens. They also propose using their measurements of wasted tokens to improve model training, reasoning strategies and harness design.

Method

To reconstruct an AI agent's activity, the researchers tracked each exchange between the model and the software controlling it, recording its reasoning, responses, tool requests and token costs – with the level of detail varying by provider: Kimi-K3 reveals its entire reasoning process; Gemini 3.1 Pro only summaries; and Claude exposes none.

Each tool's results were fed back into the agent, allowing the researchers to follow the process through to completion, or until its operating limit was reached.



Four examples of how AI agents spend tokens. Repeating known information, making identical searches, and adding pointless apologies are counted as waste, while an unsuccessful search is not necessarily wasteful, since the agent may have had good reason to try it.

The agents' requests were routed through a proxy to track their activity, with only minor code changes required, and each request tagged with its benchmark, run, task and trial (except OpenClaw, which doesn't support such tags – in that case, a separate proxy – built on LiteLLM – was used for each test).

Four Players

The study defines token waste through four recurrent patterns: *Restate* (repeating known information); *Duplicate-Fetch* (repeating unnecessary tool requests); *Padding* (adding irrelevant text); and *Detour* (pursuing plausible but ultimately unproductive approaches).

Each recorded session was examined by an LLM-as-a-Judge (LLMaJ**), which was given the original task, the complete record and criteria for identifying all four waste patterns. Suspected waste was marked and removed, and the remaining tokens recounted to calculate the wasted proportion and cost. The counts were then checked against each provider's own tokenizers or counting tools and reported output usage (with only minor formatting differences reported).

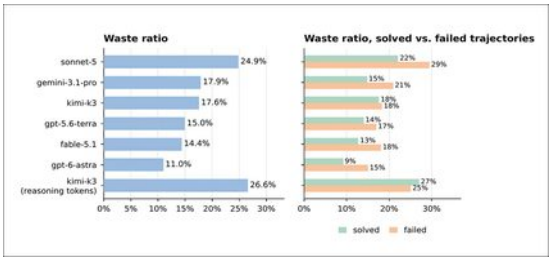
Tests

Models tested for the study comprised Kimi-K3; Gemini 3.1 Pro; GPT-5.6 Terra; GPT-6 Astra; Claude Sonnet 5; and Claude Fable 5.1, tested over the benchmarks τ²-bench (airline and retail customer service); SWE-bench Verified (software engineering); Claw-Eval; and MCP-Atlas (general-purpose tool use).

© Martin Anderson and/or commissioning entities

Additionally, Kimi-K3 was tested at different reasoning levels, and five agent harnesses compared: OpenCode; OpenClaw; OpenHands; Hermes; and Kimi Code CLI – with these latter comparisons used to assess how models, tasks, reasoning effort and harness design affect token waste (and with suspected waste removed in further tests to check its effect on performance).

To establish how much money was wasted by each model, the cost of unnecessary text and tool calls was compared against total output costs, averaging the results across the four benchmarks. Internal reasoning was assessed separately for Kimi-K3 – the only model that exposed its complete reasoning process:



Test results comparing wasted output costs across six AI models, averaged over four benchmarks. Left: overall waste rates, with Kimi-K3's internal reasoning shown separately. Right: waste rates for successful and failed tasks, with failures generally producing more waste.

As shown above, between 11% and 24.9% of the money spent generating answers and using tools was judged to have been wasted, with GPT-6 Astra performing best and Claude Sonnet 5 worst. For Kimi-K3, around one in six tokens was wasted on answers and tool use, rising to more than one in four during its internal reasoning.

More waste was generally found when tasks failed, though the reverse was true for Kimi-K3's reasoning; and there was no clear evidence that spending more tokens on unnecessary work improved the chances of success.

In another test, an attempt was made to cut wasted tokens while the AI was still working: at each step, an AI judge** was used to remove repeated information and pointless text from the model's reasoning, before the task was allowed to continue:

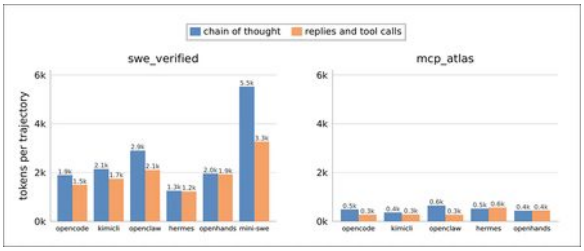
	SWE-bench Verified		MCP-Atlas	
Pruning Method	deleted (%)	solved (%)	deleted (%)	solved (%)
nothing deleted	0	**89.0**	0	**65.0**
restated and padding deleted	41	**89.5**	46	**62.3**
all reasoning deleted	100	**73.0**	100	**56.0**

Test results showing how removing unnecessary AI reasoning affected success rates on two benchmarks. Removing 41-46% of reasoning had little effect on performance, while removing all reasoning caused substantially more failures.

On SWE-bench Verified, 41% of reasoning was cut away, with no loss of performance, with the success rate actually rising slightly, from 89% to 89.5%. On MCP-Atlas, 46% was removed, with success falling only modestly, from 65% to 62.3%.

In other words, almost *half the reasoning could be discarded while the AI was working*, without seriously harming its ability to finish the job.

Finally†, Kimi-K3 was tested on identical tasks using different agent harnesses. Here, reasoning output varied more than fourfold, without any consistent improvement in performance, as shown below:



Test results comparing Kimi-K3's token consumption across agent harnesses on two benchmarks. Internal reasoning is shown in blue, and responses and tool calls in orange. On coding tasks, more than four times as much reasoning was generated with Mini-SWE as with Hermes.

Some waste was traced to basic software faults. OpenCode, Kimi Code CLI and OpenClaw launched coding tests in the wrong Python environment, causing errors and retries.

Conclusion

This is only a glance at a paper whose length and assiduous method may reward a deeper read.

© Martin Anderson and/or commissioning entities

The subject matter did bring to mind something that I am often conscious of: the sheer gulf between what we and the frontier suppliers actually understand about these processes, and the companies' willing disposition to package the product as if it had just spent the last three months in a linter, and passed with flying colors.

As it stands, for the moment, for bad outcomes, we can currently blame no-one but ourselves, because we are buying what nearly all of us know to be a deeply experimental and unpredictable product.

* *With avowedly equal contributions from University of Illinois Urbana-Champaign, Northeastern University, University of California (Los Angeles), and an independent researcher.*

* *The paper does not identify which model was used, as far as I can discern.*

† *For this article, not the paper, which is longer than I usually permit myself to cover, and contains more tests and depth than we can accommodate here. Please refer to the source PDF, linked above, for more details.*

First published Friday, November 6, 2026

Writer on machine learning, domain specialist in human image synthesis. Former head of research content at Metaphysic.ai, until its dissolution into DNEG's Brahma.ai.
Website: martinanderson.ai
Contact: martin@martinanderson.ai