DECIDING WITHOUT GENERATING: WHEN DOES A STANDALONE DECISION MODEL PAY OFF IN LLM SYSTEMS?

Ahmed Elmersawy Amar AlAzizy

ABSTRACT

Much of what a language model does inside a production system is not writing but deciding: which tool to call next, whether a cached answer can be reused, which model should answer. A new class of non-generative "decision models" returns a typed choice with a probability per option instead of text and is marketed as a far cheaper replacement for an LLM at these points. We test that claim on one harness with interchangeable deciders (a frontier LLM, small hosted and self-hosted LLMs, a proprietary decision model, an open-weights reproduction of it, and trained classifiers) at three decision points, measuring quality, cost, latency and GPU energy. The decision model does not pay off for agent tool choice, where letting the generator decide inside its own call is cheapest at equal task reward. It pays off as a semantic-cache guard, which decides whether a stored answer can be served for a new query: on math questions whose near-identical variants may change the answer, it decides correctly 93.3% of the time, against 81.7% for a classifier trained on that dataset and 66.1% for a similarity threshold, at about 2% of the frontier model's cost per decision. For model routing, a small classifier trained on the cheap model's record beats every general decider. Across the three decision points, a separate decider was worth its call only when the decision avoids a generation and the evidence it needs is in the input.

1 INTRODUCTION

A deployed LLM system makes many small decisions around each answer it writes. An agent decides which tool to call next, or whether to stop and reply (Yao et al., 2024). A semantic cache decides whether a stored answer can be served instead of a new one (Bang, 2023; Schroeder et al., 2025). A router decides which model should handle a prompt (Chen et al., 2023; Ong et al., 2024). Each of these decisions has a small, fixed set of outcomes (which tool, reuse or regenerate, which model), yet the usual way to make one is to ask a generative model and parse its text: the system pays for a full autoregressive call when all it needs back is one choice among a handful of options.

A recent class of models targets exactly this gap. These *decision models*, which their vendors call "System One" models after the fast mode of dual-process theory (Kahneman, 2011), are not autoregressive: given a state and a typed question (pick one of these options, score on this scale, yes or no), they return the answer with a probability for each option in a single pass. The first commercial one, Jev (TypeSafe AI, 2026), is priced at a small fraction of a frontier LLM per input token with no output cost, and is advertised as a drop-in replacement for an LLM wherever the output is a decision. An open-weights reproduction, Kev (Palmer, 2026), followed within weeks.

Whether this replacement pays is an empirical systems question, and it has two parts that vendor numbers do not separate. The first is quality: is the cheap decision as good as the expensive one? The second is structure: what does the decision buy? A cache guard's decision can remove a generation entirely, so a cheap, accurate guard saves nearly the whole call. An agent's tool choice sits in front of a generation that happens anyway, because some model must still write the tool's arguments, so a separate decider adds a call instead of removing one.

We study both parts on one harness. Every decider implements the same interface (a state, a set of options, a returned choice with a confidence), so the same code path scores a frontier LLM, a small hosted LLM, a self-hosted LLM, Jev, Kev, and trained classifiers. We run them at three decision points on public datasets, with repeated calls, clustered confidence intervals, measured cost per call, and measured GPU energy for the self-hosted deciders.

Our findings are as follows.

Agent control: the decision model does not pay off. τ^2-bench (Barres et al., 2025) simulates airline customer-service conversations in which an agent must look up and change bookings through tools while following a written policy. At saved points in such conversations, Jev picks the

Correspondence to: Ahmed Elmersawy <ahmedalmersawy81@gmail.com>.

Preprint.

same next action as a frontier model's successful run 81.4% of the time, at $1/22$ of the frontier model's cost per decision and median latency. Inside a live agent loop, however, the standard design, in which the generating model decides and acts in one call, is the cheapest of all and matches the best reward. Every split design costs more per success, and with Jev deciding it also loses about five points of reward, a gap at the edge of significance.

Cache guard: the decision model pays off. GSM-Plus (Li et al., 2024b) rewrites grade-school math questions into variants: some only reword the question and keep its answer, others change a number and so change the answer. Used as cache traffic, a new question and its nearest cached neighbor often differ by that one number, so embedding similarity cannot tell a safe reuse from a wrong one, and the best similarity threshold, tuned on held-out queries, reuses 0.3% of queries (66.1% accuracy). Jev reaches 93.3%, within 2.6 points of the frontier model, at about 2% of its cost per decision. On GSM-Plus a frontier model used as the guard is self-defeating: paying for its decision on every query raises total cost to 138% of running without a cache. On LMArena (chat prompts) and SearchQueries (web search queries), two datasets released with vCache (labels corrected, Appendix A), Jev ties the similarity threshold on chat, where similarity works, and beats it by 4.4 points on search, where it does not. A cascade that uses the free similarity score for clear cases and calls Jev only when the score is ambiguous is best or tied for best on the two traffic datasets and matches Jev on GSM-Plus.

Routing: a trained classifier wins. RouterBench (Hu et al., 2024) records which prompts a small, cheap model (Mixtral-8x7B) and a large, expensive one (GPT-4) answered correctly. Asked directly to send each prompt to one of the two, no general decider, frontier LLM included, does better than a random mix from the prompt alone, and as rankers they add at most 3.0 points of quality over a random mix. A small classifier trained on the cheap model's record of right and wrong answers beats all of them, and even it adds only 4.5 points.

The pattern. Across the three points, a standalone decider was worth its call only when two conditions held: the decision avoids a generation, and the evidence it needs is visible in its input. Agent control fails the first, cache guarding meets both, and routing fails the second.

We make two contributions: (1) an independent measurement of decision models at three decision points, with per-call cost and latency at the agent and cache points and GPU energy for local deciders, against open baselines that need no API; and (2) a harness that makes a typed-output decision model interchangeable with LLMs and classifiers.

2 Background

Decision models. A decision model maps a textual *state* and a typed question to a structured decision, with a probability per option. In this work, we focus on choice questions: each specifies a set of named options, a short description of each option, and instructions for selecting among them. Unlike a conventional language-model call that returns a generated textual response, a decision-model call returns a choice in a predefined format. This interface can avoid generating a free-form output, but it does not, by itself, guarantee lower end-to-end latency, cost, or energy: these depend on the model, serving configuration, and downstream computation.

Jev (TypeSafe AI, 2026) is a proprietary decision model accessible through an API; its weights, architecture, and training data are not publicly available. We evaluate version 1.13 and treat vendor claims about its performance as hypotheses to test empirically. Kev (Palmer, 2026) is a third-party, Apache-2.0-licensed implementation of the decision-model interface, available in four sizes from 0.8B to 27B parameters (we use 0.8B, 4B and 9B) and accepting the same API request format as Jev. It does not use Jev's weights, and we do not assume that the two models are behaviorally equivalent. We include Kev for two complementary reasons: it provides an open model that can be evaluated without access to the proprietary API, and it enables local measurements of inference latency and GPU energy that are not directly observable through a hosted API. We therefore use Jev to evaluate the proprietary service and Kev to study the behavior and serving costs of an open, locally deployed alternative.

Three decision points. We study three places where a standalone decision model could sit in an LLM system: agent control, semantic-cache guarding and model routing (Table 1). They differ in whether the decision can remove a generation. In agent control it cannot: the decision picks a tool or a reply, but some model must still write the tool's arguments or the reply, so a separate decider adds a call. A cache guard's decision to reuse skips the generation, and a router's decision to stay with the small model skips the large one's. At those two points the decider must still beat a cheaper alternative, a similarity threshold or a trained classifier, to justify its own call.

3 Harness and Method

3.1 One interface, interchangeable deciders

A decider receives a rendered state, an instruction, and a set of options, and returns one option with a confidence. Every decider at a decision point sees the identical rendered state and the identical option descriptions. We evaluate the following deciders.

Table 1. The three decision points studied in this work. "Replaces generation?" indicates whether a decision can avoid a subsequent generative-model call. "Cheap alternative" identifies the baseline against which the standalone decider must justify its additional cost.

Decision point	Options	Dataset (test size)	Replaces generation?	Cheap alternative
Agent control	next tool, or reply	τ^2-bench airline (1,076 states; 50 tasks)	No	generator's own call
Cache guard	reuse / regenerate	GSM-Plus, LMArena, SearchQueries (2,000 each)	Yes	similarity threshold
Router	small / large model	RouterBench (1,998 prompts)	Yes	trained classifier

Decision models. Jev 1.13 through its hosted API; Kev (4B by default, also 0.8B and 9B for agent control) served locally.

LLMs. GPT-5.6 as the frontier model; GPT-OSS-20B as a small hosted LLM; Qwen3-8B (Yang et al., 2025) served locally with vLLM (Kwon et al., 2023), with thinking turned off. An LLM decides through one constrained call: the options are the only permitted outputs (a forced tool choice), and its confidence is the one it states.

Baselines specific to a decision point. For the cache guard: a threshold on the embedding similarity of the two queries; an off-the-shelf duplicate-question cross-encoder (Reimers & Gurevych, 2019); vCache (Schroeder et al., 2025), which learns a similarity threshold per cached prompt under a target error rate δ; and a classifier fine-tuned on each dataset's dev split. For the router: a classifier fine-tuned on the small model's record of right and wrong answers, plus three fixed policies (always small, always large, and a random mix of the two).

Cascades. A cheap first stage answers when it is confident and escalates otherwise: Jev to GPT-5.6 (agent control), and similarity threshold to Jev (cache guard). Every cascade threshold or band is chosen on data the reported number does not use.

3.2 Measurement

Repeats and intervals. Stochastic deciders are called five times per item (GPT-5.6 three times at the cache guard and router, to bound cost). All intervals are 95% cluster bootstraps: over states or tasks for agent control, over classes for the cache guard, over prompts for the router. Comparisons between two deciders are paired on the same items.

Cost. For hosted models we record the cost the API gateway reports for each call, not a token count multiplied by a list price; the two differed by about $3\times$ for GPT-OSS-20B because of upstream provider selection. All hosted calls go through one gateway and one key. Calls from different deciders are interleaved so that network conditions affect them equally.

Latency. Wall-clock time per call at the client. For Kev we also record server-side model time, which isolates the median HTTP overhead (7–8 ms).

Energy. For local deciders we read the GPU's cumulative energy counter (NVML) around a contiguous block of calls on a dedicated A100-80GB, with one request in flight, prefix caching off, and warm-up calls discarded. We report gross joules per decision and the value net of the idle baseline. The counter is validated against the instantaneous power reading before each run; on one consumer GPU it failed that check, and we report no energy for that machine. Hosted deciders have no energy number.

Failures. A call that fails for an infrastructure reason (billing, rate limit, provider error, network) is retried and never counted against the decider. A call in which the decider produces no valid choice is a decider failure, counted as wrong and never retried, so a stochastic decider cannot re-roll its own failures.

Calibration. Expected calibration error (ECE) (Guo et al., 2017) over the probability of the chosen option.

4 Agent Control

Task. At each agent turn, choose the next tool or choose to reply to the user. We use the airline domain of τ^2-bench (Barres et al., 2025): 50 tasks (booking, changing and cancelling flights) in which an LLM-simulated user talks to the agent and the benchmark grades the outcome. We pin release v1.0.1.

We run two experiments. The first scores the decision in isolation; the second places each decider inside the agent loop.

4.1 Decision-only replay

We collect agent states from GPT-5.6 acting as a normal single-call agent (50 tasks $\times$ 3 episodes) and keep the 1,076 states from episodes that succeeded. Each decider is then asked, at each saved state, for the next action; nothing is executed. The label is the action the reference agent took. This is a lower bound on quality, since other valid actions exist.

Table 2 shows that Jev agrees with the reference on 81.4% of states [79.1, 83.6], 7.1 points below GPT-5.6 [5.2, 9.1], while being 22 times cheaper and 22 times faster at the median, and the best calibrated decider (ECE 0.029). It is 14 points more accurate than the small hosted LLM at about

Table 2. Decision-only replay on 1,076 airline states, 5 repeats. Accuracy = agreement with the reference action; local = self-hosted, no API cost. †Graded against its own trajectories, so an upper bound.

Decider	Acc. (%)	p50 (ms)	$/decision	ECE
GPT-5.6†	88.5	3,909	3.83×10^{-3}	0.103
Jev → GPT-5.6	84.9	162	8.16×10^{-4}	0.063
Jev	81.4	176	1.73×10^{-4}	0.029
GPT-OSS-20B	67.0	1,299	3.11×10^{-4}	0.286
Kev-9B	40.1	548	local	0.179
Qwen3-8B	39.6	656	local	0.599
Kev-4B	32.1	439	local	0.124
Kev-0.8B	27.3	218	local	0.088

half its cost. The vendor's claim of a cheaper, faster decision therefore holds for the single decision, at a measured quality cost.

A cascade that escalates to GPT-5.6 when Jev's confidence is below 0.65 (a threshold chosen on held-out tasks) recovers half of the gap: 84.9% [82.8, 86.9], with 16.7% of calls escalated, at 21% of GPT-5.6's cost and Jev's median latency.

The open deciders are far behind on this domain. Kev reaches 27–40% depending on size and Qwen3-8B 39.6%. Kev also rejects 34 states (35 for Kev-0.8B; 3%) that exceed a hard input limit in its encoder; we count these as failures. Qwen3-8B is almost always certain (mean confidence 0.996) and wrong on 60% of states.

Energy. At similar accuracy (Kev-9B versus Qwen3-8B), Kev-9B uses about 20% less energy and is 17% faster; the gap is far from the order-of-magnitude difference seen in API costs. On the same A100-80GB, a decision costs 47 J with Kev-0.8B, 132 J with Kev-4B, 170 J with Kev-9B and 211 J with Qwen3-8B (gross; 29, 100, 130 and 167 J net of idle).

4.2 In the agent loop

A decision model cannot write tool arguments or a reply, so in a real agent a generator G must act after every decision. We compare five designs, which we call arms, on the same tasks, with the same user simulator and seeds, and G = GPT-5.6:

A: Jev decides, then G executes the chosen action. **B**: G decides in a constrained call, then G executes. **C**: G decides and executes in a single call, the standard agent. **D**: GPT-OSS-20B decides, then G executes. **E**: Kev-4B decides, then G executes.

If a decider fails on a turn, G decides that turn (fallback rate: A 0%, B 0.7%, D 8.7%, E 6.3%). Each arm runs 50 tasks × 5 trials.

Table 3. Airline tasks in the agent loop (generator G = GPT-5.6, 250 episodes per arm; 6 E episodes that crashed when G returned an empty message count as failures). Reward and pass^5 in %, reward with 95% CI over tasks; pass^5 = all five trials of a task succeed; decision share = decider's part of the agent's cost (E's decider is local).

Arm	Reward	pass^5	$/success	Dec. share
A: Jev + G	74.4 [63.6, 84.4]	60.0	0.048	5.3%
B: G + G	79.6 [69.6, 88.8]	66.0	0.176	74.5%
C: G alone	79.2 [69.6, 88.4]	64.0	0.033	0.0%
D: GPT-OSS + G	69.2 [57.2, 80.4]	54.0	0.043	4.9%
E: Kev-4B + G	4.4 [1.2, 8.8]	0.0	0.476	0.0%

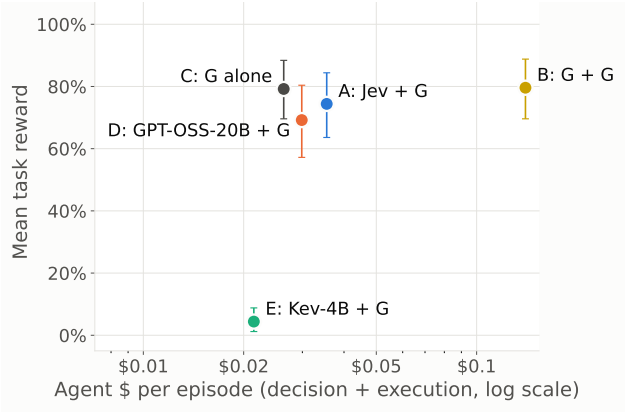


Figure 1. Task reward against agent cost per episode for the five arms (log cost axis; error bars are 95% CIs over tasks). The single-call agent C is cheapest at the top reward.

Table 3 and Figure 1 show that the single-call agent (C) is the cheapest arm per success and ties for the best reward. Asking G for the decision separately (B) changes reward by $+0.4$ points $[-2.8, +3.6]$ and adds \$0.114 of cost per episode, because the long conversation is now sent twice per turn. Replacing that decision with Jev (A) removes almost all of the added cost, so A is \$0.105 per episode cheaper than B, but A still costs \$0.009 more per episode than C and loses 4.8 points of reward against it $[-10.8, +0.8]$, and 5.2 points against B $[-10.8, 0.0]$, at the edge of significance. A's reward loss is close in size to Jev's 7.1-point accuracy gap in Table 2, which suggests the single-decision gap carries into the loop.

Arm E fails: Kev-4B asks for user details before the user has given an identifier, and repeats the call until the episode is cut off. The same behavior appears with a local Qwen3-8B as G (reward 4.8%), so it belongs to the decider and not to the generator.

With a weak generator (G = Qwen3-8B, all local), separating the decision helps: arm B reaches 30.8% against 19.6% for C, a paired gain of 11.2 points [3.2, 20.0]. A split may

help a weak generator structure its work; it does not help a frontier one.

Takeaway. With a frontier generator after every decision, the decision is nearly free inside that generation (B costs 5 times as much per success as C), and every separate decider we ran added cost or lost reward.

5 CACHE GUARD

Task. A semantic cache has found the stored prompt nearest to a new query. Should it *reuse* the stored answer or *regenerate*? A correct reuse skips the model call; a wrong reuse serves a wrong answer.

Scope of the decision. Deciders see the new query and the cached query, and never the cached answer or the similarity score. The decision is "are these the same request?". Whether the stored answer was right belongs to the model that wrote it and to the cache's expiry policy, and reading long answers would cost much of what the cache saves. An earlier version of our question asked deciders to check the stored answer; the frontier model then regenerated on 24% of LMArena pairs whose two prompts were *identical*, because it disagreed with the stored answer. We discarded those runs.

5.1 Data

We use three public datasets and build nothing ourselves, so that we cannot choose the hard cases. Each is a stream: the cache holds every earlier prompt, and each query's candidate is its nearest cached neighbor under GTE-large (Li et al., 2023b), an open text-embedding model, the same for every method. Each dataset is split by class into 600 dev queries (for tuning thresholds, bands and the trained classifier) and 2,000 test queries.

GSM-Plus (Li et al., 2024b): variants of grade-school math questions from GSM8K (Cobbe et al., 2021), for example a rewording, an added distractor, or a changed number; the original questions fill the cache. Reuse is correct for 33.6% of queries. Labels are exact numeric answers. Here wrong reuses are *more* similar to their neighbor than correct ones (median cosine 0.966 against 0.955), so no threshold can separate them.

LMArena and **SearchQueries**: two datasets released with vCache (Schroeder et al., 2025). LMArena takes user prompts from Chatbot Arena (Chiang et al., 2024) and adds LLM-written rewordings of each; SearchQueries takes real web search queries from Bing's ORCAS log (Craswell et al., 2020) and groups those that ask the same thing. Their labels assign each query a *class* and count any two queries of a class as a correct reuse, so nobody judged the pair a cache actually meets. We relabeled each pair directly with an LLM (Claude Opus 5.5) under a written guideline fixed in advance, which changed 6.0% of LMArena and 15.7% of SearchQueries labels (Appendix A). On the corrected labels, which all results below use and which we release with the reason for each change, reuse is correct for 92.5% and 58.2% of queries.

5.2 Results

Table 4 and Figure 2 give the results.

Where similarity is blind, the decision model wins by a wide margin. On GSM-Plus the dev-tuned threshold, the cross-encoder and vCache all reuse at most 1.3% of queries, because no threshold gives acceptable precision; their accuracy is the regenerate-always rate. Jev reaches 93.3%, which is 27.2 points above the threshold [25.5, 29.1] and 11.7 points above a classifier trained on this dataset's own dev split [10.1, 13.3], with a 1.1% wrong-reuse rate. GPT-5.6 is 2.6 points better [1.9, 3.4] at 48 times the cost per decision.

On the two traffic datasets, the decision model ties similarity or beats it. On LMArena the threshold and Jev are statistically tied on accuracy ($+0.6$ for the threshold $[-0.4, +1.6]$), and Jev serves fewer wrong answers (-1.7 points $[-2.3, -1.2]$). On SearchQueries Jev is 4.4 points more accurate [2.3, 6.4] and cuts wrong reuses from 14.4% to 3.8% $[-12.1, -9.1]$. GPT-5.6 is not better than Jev on either dataset (-0.4 points $[-1.5, +0.8]$ on SearchQueries); on LMArena it is the weakest LLM decider, because it regenerates 8.4% of all queries although reuse was correct. vCache keeps its error bound on every dataset, and pays for it on the two traffic datasets by regenerating 45–48% of all queries although reuse was correct.

A cascade gets the best of both. Calling Jev only when the similarity falls in an ambiguous band chosen on dev (3% of LMArena queries, 72% of SearchQueries, all of GSM-Plus) is best or tied for best on the two traffic datasets and matches Jev on GSM-Plus, and on LMArena costs $\$6 \times 10^{-7}$ per query.

The open decision model is competitive here. Unlike in agent control, Kev-4B is within a point of Jev on the two traffic datasets and 5.2 points behind on GSM-Plus [4.0, 6.6], at 70 ms per decision on a local GPU, though it serves 1.6 to 4.8 times as many wrong answers.

5.3 What the decision is worth

A guard only pays if its decision is much cheaper than the answer it saves. We measured the mean cost per query of regenerating with GPT-5.6 on 200 queries each of GSM-Plus and LMArena (SearchQueries has no answers): \$0.0016 and \$0.0084. The expected cost of a query is then the decision's cost plus the regeneration cost times the share of queries

Table 4. Cache guard on 2,000 test queries per dataset. Acc. = share of correct decisions; WR = wrong reuses as a share of all queries, both in %. Thresholds are tuned for accuracy on dev. $ and p50 latency are per decision, averaged over the three datasets; local = self-hosted, no API cost; free = no model call; – = not measured. vCache runs unmodified as a whole system: it learns its thresholds online from the datasets' original class labels and caches only the prompts it missed, so its row is a reference point and not a like-for-like decider.

	GSM-Plus		LMArena		SearchQueries			
Decider	Acc.	WR	Acc.	WR	Acc.	WR	$/decision	p50 (ms)
Jev	93.3	1.1	96.3	1.2	85.6	3.8	2.1×10^{-5}	180
Threshold → Jev	93.3	1.1	97.4	2.5	85.7	3.8	1.2×10^{-5}	–
GPT-5.6	96.0	0.9	91.2	0.4	85.2	1.5	1.0×10^{-3}	1,250
GPT-OSS-20B	95.0	1.3	93.3	1.0	83.6	3.4	4.1×10^{-5}	1,310
Kev-4B (local)	88.1	5.3	96.9	1.9	85.4	8.6	local	70
Qwen3-8B (local)	86.3	2.5	95.7	1.8	84.8	6.5	local	390
Trained classifier	81.7	2.9	95.7	2.5	77.1	12.6	local	–
Similarity threshold	66.1	0.3	96.9	2.9	81.2	14.4	free	–
Cross-encoder	66.3	0.3	96.5	3.3	78.4	17.1	local	–
vCache ($\delta = 0.05$)	66.1	0.7	54.6	0.3	50.3	1.9	free	–

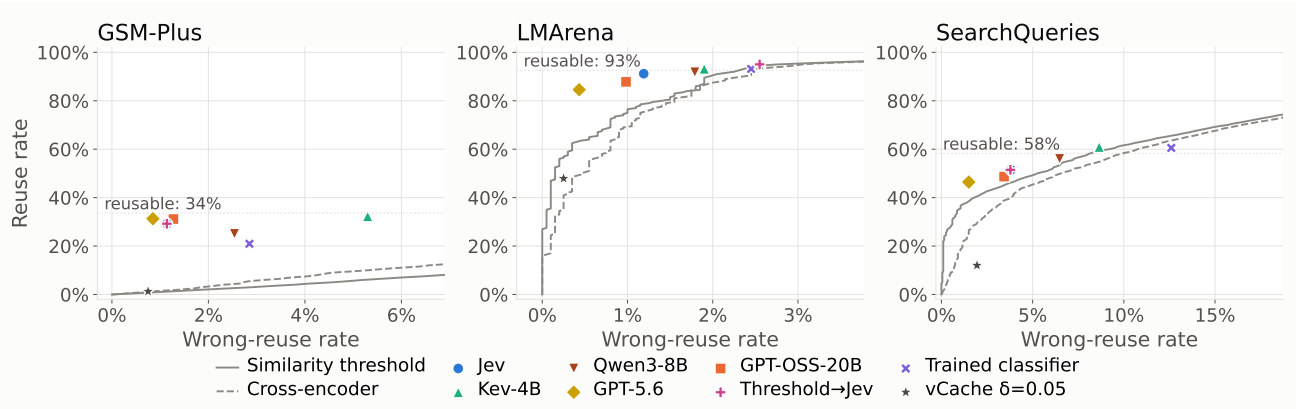


Figure 2. Reuse rate against wrong-reuse rate. Lines sweep the similarity threshold and the cross-encoder score; each decider is one point; the dotted line is the share of queries for which reuse is correct. Up and to the left is better; each x-axis ends just past its rightmost decider. On GSM-Plus every learned decider sits far above both curves, and Threshold→Jev coincides with Jev.

regenerated.

On GSM-Plus, where only a third of queries can be reused, Jev as the guard brings the cost to 72% of running without a cache; the similarity methods stay at about 100%. GPT-5.6 as the guard raises it to 138%: the decision alone costs $0.0011, two thirds of the answer it is deciding about. On LMArena, where most queries can be reused, the free threshold brings the cost to 5% of no-cache, the cascade to 5%, Jev on every query to 9%, and GPT-5.6 to 28%. These figures assign no cost to a wrong answer; any positive cost for one favors Jev and the cascade further over the threshold, since on LMArena and SearchQueries they serve fewer.

Takeaway. When one outcome of the decision skips a generation and the two queries hold all the evidence, a cheap decision model is the guard to use: it ties or beats similarity on every dataset, while a frontier model's decision can cost two thirds of the answer it may save.

6 ROUTER

Task. Send a prompt to a small model or a large one. As with the cache guard, the decision replaces a generation: the expensive call is skipped when the small model would have been right.

Data. RouterBench (Hu et al., 2024) records graded answers and API costs of eleven models on the same prompts, so quality and model cost need no new calls. We use the pairing of the routing literature (Ong et al., 2024), Mixtral-8x7B as the small model and GPT-4 as the large one, on the six task families whose answers are graded exactly (MMLU, GSM8K, ARC, HellaSwag, WinoGrande and MBPP), stratified by family into 1,998 test and 594 dev prompts, with the remaining 32,106 for training. The small model scores 0.589 on the test set and the large one 0.792; 28% of prompts need the large model.

Deciders. Each decider sees the prompt only, is told which

Table 5. Routing on 1,998 RouterBench prompts. "Own choice" = share sent to the large model when the decider simply answers. Gain = quality over a random mix sending the same share, at the dev-tuned operating point (30% of dev prompts sent to the large model; 23–30% on test), in points. Qwen3-8B and GPT-5.6 state few distinct confidences (the largest tie covers 1,061 and 1,404 prompts); the point estimate breaks ties in prompt order and the bootstrap at random, so their APGRs sit near an edge of the CI (random tie-breaking gives 0.516 and 0.537).

Router	APGR [95% CI]	Own choice	Gain
Trained classifier	0.649 [0.621, 0.678]	3.9%	+4.5
Jev	0.609 [0.585, 0.642]	0.0%	+2.7
Kev-4B	0.582 [0.553, 0.609]	0.0%	+3.0
GPT-OSS-20B	0.546 [0.516, 0.576]	4.7%	+1.8
Qwen3-8B	0.531 [0.488, 0.540]	20.6%	−1.1
GPT-5.6	0.529 [0.510, 0.568]	2.3%	+1.2
Random mix	0.500	–	0.0

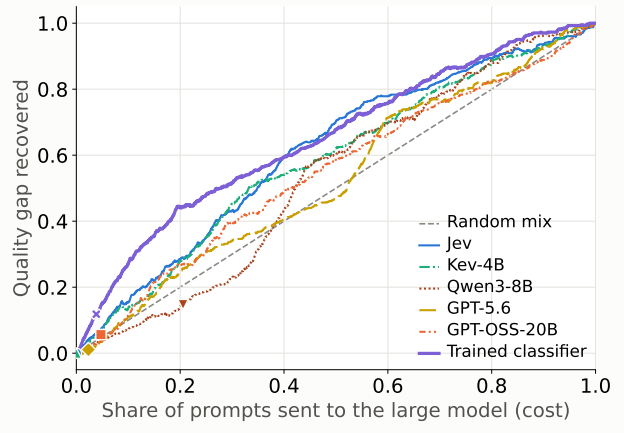


Figure 3. Share of the small-to-large quality gap recovered against the share of prompts sent to the large model. The dashed diagonal is a random mix. Markers show each decider's own choice.

two models it is choosing between, and returns small or large with a probability. The trained baseline is a DistilRoBERTa classifier (Sanh et al., 2019) fine-tuned on 20,000 prompts sampled from the training split to predict whether the small model is wrong and the large one right.

Metrics. A decider's own choice gives one cost-quality point. Sweeping a threshold on its probability gives a curve, summarized by APGR, the average performance gap recovered (Ong et al., 2024): the share of the small-to-large quality gap recovered, averaged over all shares of prompts sent to the large model. We also report one operating point fixed in advance: the threshold at the 70th percentile of each router's dev scores, applied unchanged to test.

Results. Table 5 and Figure 3 show three things.

First, asked directly, no general decider routes better than a random mix. Jev and Kev send nothing to the large model, and GPT-5.6 sends 2.3%. At those points, quality is within 1.6 points of a random mix sending the same share.

Second, as rankers the general deciders carry a weak signal, and the frontier model is among the weakest. Jev's probabilities rank prompts best among the general deciders (APGR 0.609), ahead of GPT-5.6 by 0.080 [0.045, 0.106].

Third, the trained classifier beats every general decider, Jev included, by a small margin: +0.040 APGR [+0.001, +0.070]. At the fixed operating point it gains 4.5 points of quality over a random mix [3.3, 5.8], against 2.7 for Jev [1.5, 3.9]. Even so, its APGR is only 0.149 above a random mix, which matches the routing literature's experience that learned routers are hard to separate from a random mix.

Why. The routing question is "will *this* small model get *this* prompt right?". The answer likely depends on that model's particular strengths, which the prompt does not show and a general decider has no way to know. A classifier trained on the model's record has seen that information; a decision model reading the prompt has not.

Takeaway. Avoiding a generation is not sufficient. The decider must also be able to see what the decision depends on.

7 Discussion

Two conditions. Across our three decision points, a standalone decider paid for its call only when both of the following held.

The decision avoids a generation. In agent control a generator runs after every decision, so letting that generator decide is the cheapest design in our experiments. At the cache guard and the router, one outcome of the decision skips the expensive call.

The evidence is in the input. Whether two queries ask the same thing can often be judged from the queries. Whether a particular small model will answer a prompt correctly is harder to infer from the prompt alone.

In our experiments, agent control fails the first condition; the cache guard meets both and is where the decision model pays off; routing remains difficult because correctness depends on model behavior. These conditions may also help reason about other decision points: input guardrails and intent classification can meet both, while "should the agent retry?" may not, since a generation follows it anyway.

The decision must be much cheaper than what it saves. A frontier LLM was the most accurate guard on GSM-Plus and still raised total cost to 138% of running without a cache. The relevant quantity is the cost of a decision relative to the generation it may avoid, weighted by how often it avoids

one.

What the vendor claims amount to. In our runs, Jev cost 22 to 48 times less per decision than the frontier model, and its median latency was about 180 ms. Its quality was competitive at the cache guard and lower than the frontier model for single agent decisions (7.1 points below). Our results do not establish that it can replace an LLM at every decision: in the agent loop and at the router, a cheaper or better alternative existed. Its probabilities were the best calibrated in the agent replay (ECE 0.029, Table 2), which the agent cascade relies on.

Open decision models. Kev shows that the paradigm is reproducible but not uniformly effective: it came within a point of Jev on judging whether two traffic queries are the same request (5.2 points behind on GSM-Plus) and reached 27–40% replay accuracy on multi-step airline policy. On equal hardware, Kev-9B's energy advantage per decision over a same-size generative LLM making a constrained choice was about 20%, not an order of magnitude. A constrained LLM call emits a median of 32 tokens against about 3,400 input tokens, so eliminating output generation alone may yield limited energy savings.

Limitations. Jev is proprietary and versioned; our numbers describe version 1.13 through one gateway in September–October 2026. Agent control uses one domain, with labels and a generator from one frontier model. The LLM-revised labels of the two vCache datasets were not checked by hand. Hosted deciders have cost and latency measurements but no energy. RouterBench's models are from 2023, and the router predicts those models' recorded correctness; results may differ for current models. We did not fine-tune Kev or any LLM for a decision point; a tuned decision model might close the routing gap, but would need comparison against a trained classifier.

8 Related Work

Decision points in compound systems. LLM applications are increasingly built as systems of models, retrievers and tools rather than as one model call (Zaharia et al., 2024), and each junction in such a system is a decision. Prior work optimizes each junction on its own terms; we compare one class of component, a standalone decider, across three junctions on one harness and ask when it earns its call.

Semantic caching. GPTCache (Bang, 2023) serves a stored answer when the embedding similarity of two prompts passes a threshold. Later systems improve the matcher: MeanCache (Gill et al., 2025b) trains the similarity model with federated learning and tracks conversational context, LangCache-style embedders are fine-tuned on domain paraphrase pairs (Gill et al., 2025a), ensembles combine several embedding models (Ghaffari et al., 2025), and SCALM (Li et al., 2024a) clusters real chat logs to decide what to cache. vCache (Schroeder et al., 2025) learns a threshold per cached prompt with a user-set error bound. All of these still decide a hit by similarity, which recent work shows is not the same as validity (Zhang et al., 2026). Closest to our guard, Krites (Singh et al., 2026) sends near-miss neighbors to an LLM judge, off the serving path to hide its latency. We evaluate a judge that reads both queries inline and find that a non-generative decision model is cheap and fast enough for that role, while a frontier LLM can cost more than the answers it saves.

Model routing and cascades. FrugalGPT (Chen et al., 2023) cascades models with a learned scorer, AutoMix (Aggarwal et al., 2024) lets a small model verify itself, and Hybrid LLM (Ding et al., 2024), RouteLLM (Ong et al., 2024) and RouterDC (Chen et al., 2024) train routers on quality or preference labels. Shnitzer et al. (Shnitzer et al., 2023) frame routing as per-model correctness prediction from benchmark records, the recipe our trained baseline follows, and Dekoninck et al. (Dekoninck et al., 2025) unify routing and cascading and identify the quality estimator as the deciding factor. Cascade deferral by confidence is well understood for classifiers (Jitkrittum et al., 2023) and has been extended to token-level uncertainty in LMs (Gupta et al., 2024). Benchmarks from RouterBench (Hu et al., 2024) to RouterEval (Huang et al., 2025), RouterArena (Lu et al., 2025) and LLMRouterBench (Li et al., 2026) standardize router comparison, and evaluations repeatedly find a large gap to the oracle and routers that struggle to beat simple baselines (Li et al., 2026): a tuned kNN matches complex learned routers (Li, 2025), many routers level off at a common plateau (Lu et al., 2026), and the small model's own uncertainty often beats a separate router (Ramírez et al., 2024); see (Varangot-Reille et al., 2026) for a survey. We add general, untrained deciders to this comparison: none replaces a classifier trained on the small model's record, and even that classifier reaches an APGR of only 0.649, against 0.5 for a random mix.

Tool use and split agents. Tool-using LLMs decide when and which tool to call inside their own generation (Schick et al., 2023; Patil et al., 2024; Qin et al., 2024), and are evaluated on BFCL (Patil et al., 2025) and the τ-bench family (Yao et al., 2024; Barres et al., 2025). A line of work splits the agent into roles: a planner and workers that avoid re-reading observations (Xu et al., 2023), a planner, caller and summarizer built from small models (Shen et al., 2024), a planner that emits a DAG of parallel calls (Kim et al., 2024), a separately trained planner for long-horizon tasks (Erdogan et al., 2025), and small on-device models that handle function calling end to end (Erdogan et al., 2024; Chen & Li, 2024), a direction recently argued for in general (Belcak et al., 2025). These splits pay through parallelism, shorter contexts or weak executors. Our in-loop arms

test the narrowest split, a per-turn tool decision in front of a frontier generator, and find it never beats the single call; with a weak generator it helps, consistent with (Shen et al., 2024). Following (Kapoor et al., 2025), we report agent results as cost against reward.

Getting a discrete decision from an LLM. Constrained decoding makes generated choices well formed (Willard & Louf, 2023; Zheng et al., 2024; Dong et al., 2025), but the model still prefills the full context and decodes, and format restrictions can cost accuracy (Tam et al., 2024). Fine-tuned small classifiers still beat zero-shot generative models on text classification (Bucher & Martini, 2024), and a classification head on an LLM backbone can replace generation for labels (Li et al., 2023a), the design Kev follows with a pointer head over a variable option set. Jev and Kev package this as a general decision model; to our knowledge this is the first independent measurement of either across decision points with cost, latency and energy.

Calibration of stated confidence. LMs are fairly well calibrated in their token probabilities on multiple-choice questions (Kadavath et al., 2022), and verbalized confidence can be trained (Lin et al., 2022) or elicited (Tian et al., 2023), but off-the-shelf verbalized confidence is typically overconfident (Xiong et al., 2024) and good uncertainty tends to need training (Kapoor et al., 2024). We see the same: the small LLMs' stated confidences are the worst calibrated in agent control (ECE 0.286 and 0.599), while the decision model's probabilities are the best (0.029).

Energy of inference. Per-query energy has been measured for generative LLM inference (Samsi et al., 2023), across task-specific and general-purpose models (Luccioni et al., 2024), with tools and benchmarks built on NVML energy counters (You et al., 2023; Chung et al., 2025), and at production scale with full-stack accounting (Elsworth et al., 2025); serving systems lower it by reconfiguring clusters (Stojkovic et al., 2025). Sampled power readings can miss much of a short run (Yang et al., 2023), so we integrate the cumulative counter and report gross and idle-subtracted energy per decision.

Label errors and LLM relabeling. Label errors are common in test sets and can change model rankings (Northcutt et al., 2021), including in LLM benchmarks (Gema et al., 2025; Vendrow et al., 2025). LLMs can match or beat crowd workers as annotators (Gilardi et al., 2023), but agreement with humans varies by task (Zheng et al., 2023; Bavaresco et al., 2025), which is why we treat our LLM-corrected labels for the two vCache datasets as a limitation.

9 Conclusion

We measured a non-generative decision model, Jev, against LLMs and trained baselines at three decision points of an LLM system. It is a strong semantic-cache guard at about 2% of the frontier model's cost per decision, a poor trade inside an agent loop that must generate anyway, and no substitute for a trained router. The rule we draw is simple to apply before building: add a standalone decider where its decision can skip a generation and the input contains what the decision depends on, and make sure the decision costs far less than the generation it may save. The harness and the open baselines are available at `https://github.com/ahmedElmersawy/Deciding-Without-Generating`, and all run records, including the corrected labels for the two vCache datasets, at `https://figshare.com/s/97da8537a6a4e8758f1b`.

References

Aggarwal, P., Madaan, A., Anand, A., Potharaju, S. P., Mishra, S., Zhou, P., Gupta, A., Rajagopal, D., Kappaganthu, K., Yang, Y., Upadhyay, S., Faruqui, M., and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. doi: 10.52202/079017-4164.

Bang, F. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS)*, 2023.

Barres, V., Dong, H., Ray, S., Si, X., and Narasimhan, K. τ^2-bench: Evaluating conversational agents in a dual-control environment. *arXiv preprint arXiv:2506.07982*, 2025.

Bavaresco, A., Bernardi, R., Bertolazzi, L., Elliott, D., Fernández, R., Gatt, A., Ghaleb, E., Giulianelli, M., Hanna, M., Koller, A., Martins, A. F. T., Mondorf, P., Neplenbroek, V., Pezzelle, S., Plank, B., Schlangen, D., Suglia, A., Surikuchi, A. K., Takmaz, E., and Testoni, A. LLMs instead of human judges? a large scale empirical study across 20 NLP evaluation tasks. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 238–255, 2025.

Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y. C., and Molchanov, P. Small language models are the future of agentic AI. *arXiv preprint arXiv:2506.02153*, 2025.

Bucher, M. J. J. and Martini, M. Fine-Tuned 'small' LLMs (still) significantly outperform Zero-Shot generative AI models in text classification. *arXiv preprint arXiv:2406.08660*, 2024.

Chen, L., Zaharia, M., and Zou, J. FrugalGPT: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023.

Chen, S., Jiang, W., Lin, B., Kwok, J. T., and Zhang, Y. RouterDC: Query-Based router by dual contrastive learning for assembling large language models. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. doi: 10.52202/079017-2120.

Chen, W. and Li, Z. Octopus v2: On-device language model for super agent. *arXiv preprint arXiv:2404.01744*, 2024.

Chiang, W.-L., Zheng, L., Sheng, Y., Angelopoulos, A. N., Li, T., Li, D., Zhang, H., Zhu, B., Jordan, M., Gonzalez, J. E., and Stoica, I. Chatbot arena: An open platform for evaluating LLMs by human preference. In *International Conference on Machine Learning*, 2024.

Chung, J.-W., Ma, J. J., Wu, R., Liu, J., Kweon, O. J., Xia, Y., Wu, Z., and Chowdhury, M. The ML.ENERGY benchmark: Toward automated inference energy measurement and optimization. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2025.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Craswell, N., Campos, D., Mitra, B., Yilmaz, E., and Billerbeck, B. ORCAS: 18 million clicked query-document pairs for analyzing search. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, 2020.

Dekoninck, J., Baader, M., and Vechev, M. A unified approach to routing and cascading for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12987–13010, 2025.

Ding, D., Mallick, A., Wang, C., Sim, R., Mukherjee, S., Rühle, V., Lakshmanan, L. V. S., and Awadallah, A. H. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations*, 2024.

Dong, Y., Ruan, C. F., Cai, Y., Lai, R., Xu, Z., Zhao, Y., and Chen, T. XGrammar: Flexible and efficient structured generation engine for large language models. In *Proceedings of Machine Learning and Systems (MLSys)*, 2025.

Elsworth, C., Huang, K., Patterson, D., Schneider, I., Sedivy, R., Goodman, S., Townsend, B., Ranganathan, P., Dean, J., Vahdat, A., Gomes, B., and Manyika, J. Measuring the environmental impact of delivering AI at Google scale. *arXiv preprint arXiv:2508.15734*, 2025.

Erdogan, L. E., Lee, N., Jha, S., Kim, S., Tabrizi, R., Moon, S., Hooper, C., Anumanchipalli, G., Keutzer, K., and Gholami, A. TinyAgent: Function calling at the edge. *arXiv preprint arXiv:2409.00608*, 2024.

Erdogan, L. E., Lee, N., Kim, S., Moon, S., Furuta, H., Anumanchipalli, G., Keutzer, K., and Gholami, A. Plan-and-Act: Improving planning of agents for Long-Horizon tasks. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, 2025.

Gema, A. P., Leang, J. O. J., Hong, G., Devoto, A., Mancino, A. C. M., Saxena, R., He, X., Zhao, Y., Du, X., Madani, M. R. G., Barale, C., McHardy, R., Harris, J., Kaddour, J., van Krieken, E., and Minervini, P. Are we done with MMLU? In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 5069–5096, 2025.

Ghaffari, S., Bahranifard, Z., and Akbari, M. An ensemble embedding approach for improving semantic caching performance in LLM-based systems. *arXiv preprint arXiv:2507.07061*, 2025.

Gilardi, F., Alizadeh, M., and Kubli, M. ChatGPT outperforms Crowd-Workers for Text-Annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30): e2305016120, 2023. doi: 10.1073/pnas.2305016120.

Gill, W., Cechmanek, J., Hutcherson, T., Rajamohan, S., Agarwal, J., Gulzar, M. A., Singh, M., and Dion, B. Advancing semantic caching for LLMs with Domain-Specific embeddings and synthetic data. *arXiv preprint arXiv:2504.02268*, 2025a.

Gill, W., Elidrisi, M., Kalapatapu, P., Ahmed, A., Anwar, A., and Gulzar, M. A. MeanCache: User-Centric semantic caching for LLM web services. In *2025 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, 2025b.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *International Conference on Machine Learning*, 2017.

Gupta, N., Narasimhan, H., Jitkrittum, W., Rawat, A. S., Menon, A. K., and Kumar, S. Language model cascades: Token-Level uncertainty and beyond. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Hu, Q. J., Bieker, J., Li, X., Jiang, N., Keigwin, B., Ranganath, G., Keutzer, K., and Upadhyay, S. K. RouterBench: A benchmark for multi-LLM routing system. *arXiv preprint arXiv:2403.12031*, 2024.

Huang, Z., Ling, G., Lin, Y., Chen, Y., Zhong, S., Wu, H., and Lin, L. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. *arXiv preprint arXiv:2503.10657*, 2025.

Jitkrittum, W., Gupta, N., Menon, A. K., Narasimhan, H., Rawat, A. S., and Kumar, S. When does Confidence-Based cascade deferral suffice? In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Kadavath, S., Conerly, T., Askell, A., Henighan, T., Drain, D., Perez, E., Schiefer, N., Hatfield-Dodds, Z., DasSarma, N., Tran-Johnson, E., Johnston, S., El-Showk, S., Jones, A., Elhage, N., Hume, T., Chen, A., Bai, Y., Bowman, S., Fort, S., Ganguli, D., Hernandez, D., Jacobson, J., Kernion, J., Kravec, S., Lovitt, L., Ndousse, K., Olsson, C., Ringer, S., Amodei, D., Brown, T., Clark, J., Joseph, N., Mann, B., McCandlish, S., Olah, C., and Kaplan, J. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*, 2022.

Kahneman, D. *Thinking, Fast and Slow*. Farrar, Straus and Giroux, 2011.

Kapoor, S., Gruver, N., Roberts, M., Collins, K., Pal, A., Bhatt, U., Weller, A., Dooley, S., Goldblum, M., and Wilson, A. G. Large language models must be taught to know what they don't know. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Kapoor, S., Stroebl, B., Siegel, Z. S., Nadgir, N., and Narayanan, A. AI agents that matter. *Transactions on Machine Learning Research*, 2025.

Kim, S., Moon, S., Tabrizi, R., Lee, N., Mahoney, M. W., Keutzer, K., and Gholami, A. An LLM compiler for parallel function calling. In *Proceedings of the 41st International Conference on Machine Learning (ICML)*, 2024.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

Li, H., Zhang, Y., Guo, Z., Wang, C., Tang, S., Zhang, Q., Chen, Y., Qi, B., Ye, P., Bai, L., Wang, Z., and Hu, S. LLMRouterBench: A massive benchmark and unified framework for LLM routing. *arXiv preprint arXiv:2601.07206*, 2026.

Li, J., Xu, C., Wang, F., von Riedemann, I. M., Zhang, C., and Liu, J. SCALM: Towards semantic caching for automated chat services with large language models. *arXiv preprint arXiv:2406.00025*, 2024a.

Li, Q., Cui, L., Zhao, X., Kong, L., and Bi, W. GSM-Plus: A comprehensive benchmark for evaluating the robustness of LLMs as mathematical problem solvers. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics*, 2024b.

Li, Y. Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers. *arXiv preprint arXiv:2505.12601*, 2025.

Li, Z., Li, X., Liu, Y., Xie, H., Li, J., Wang, F.-l., Li, Q., and Zhong, X. Label supervised LLaMA finetuning. *arXiv preprint arXiv:2310.01208*, 2023a.

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., and Zhang, M. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023b.

Lin, S., Hilton, J., and Evans, O. Teaching models to express their uncertainty in words. *Transactions on Machine Learning Research*, 2022.

Lu, Y., Liu, R., Yuan, J., Cui, X., Zhang, S., Liu, H., and Xing, J. RouterArena: An open platform for comprehensive comparison of LLM routers. *arXiv preprint arXiv:2510.00202*, 2025.

Lu, Y., Zhang, Q., Zhang, S., Yu, Z., Wang, Z., Chen, H., and Xing, J. The routing plateau: Understanding the accuracy limits of LLM routers. *arXiv preprint arXiv:2606.07587*, 2026.

Luccioni, A. S., Jernite, Y., and Strubell, E. Power hungry processing: Watts driving the cost of AI deployment? In *Proceedings of the ACM Conference on Fairness, Accountability, and Transparency (FAccT '24)*, 2024.

Northcutt, C. G., Athalye, A., and Mueller, J. Pervasive label errors in test sets destabilize machine learning benchmarks. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS 2021)*, 2021.

Ong, I., Almahairi, A., Wu, V., Chiang, W.-L., Wu, T., Gonzalez, J. E., Kadous, M. W., and Stoica, I. RouteLLM: Learning to route LLMs with preference data. *arXiv preprint arXiv:2406.18665*, 2024.

Palmer, J. Kev: an open-weights decision model. https://github.com/jaredpalmer/kev, 2026. Commit 557598f.

Patil, S. G., Zhang, T., Wang, X., and Gonzalez, J. E. Gorilla: Large language model connected with massive APIs. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024. doi: 10.52202/079017-4020.

Patil, S. G., Mao, H., Yan, F., Ji, C. C.-J., Suresh, V., Stoica, I., and Gonzalez, J. E. The Berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML)*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392, 2025.

Qin, Y., Liang, S., Ye, Y., Zhu, K., Yan, L., Lu, Y., Lin, Y., Cong, X., Tang, X., Qian, B., Zhao, S., Hong, L., Tian, R., Xie, R., Zhou, J., Gerstein, M., Li, D., Liu, Z., and Sun, M. ToolLLM: Facilitating large language models to master 16000+ real-world APIs. In *International Conference on Learning Representations (ICLR)*, 2024.

Ramírez, G., Birch, A., and Titov, I. Optimising calls to large language models with Uncertainty-Based Two-Tier selection. In *First Conference on Language Modeling (COLM)*, 2024.

Reimers, N. and Gurevych, I. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.

Samsi, S., Zhao, D., McDonald, J., Li, B., Michaleas, A., Jones, M., Bergeron, W., Kepner, J., Tiwari, D., and Gadepally, V. From words to watts: Benchmarking the energy costs of large language model inference. In *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–9, 2023. doi: 10.1109/HPEC58863.2023.10363447.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Schick, T., Dwivedi-Yu, J., Dessi, R., Raileanu, R., Lomeli, M., Hambro, E., Zettlemoyer, L., Cancedda, N., and Scialom, T. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems 36 (NeurIPS)*, 2023. doi: 10.52202/075280-2997.

Schroeder, L. G., Desai, A., Cuadron, A., Chu, K., Liu, S., Zhao, M., Krusche, S., Kemper, A., Zaharia, M., and Gonzalez, J. E. vCache: Verified semantic prompt caching. *arXiv preprint arXiv:2502.03771*, 2025.

Shen, W., Li, C., Chen, H., Yan, M., Quan, X., Chen, H., Zhang, J., and Huang, F. Small LLMs are weak tool learners: A multi-LLM agent. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 16658–16680, 2024. doi: 10.18653/v1/2024.emnlp-main.929.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Large language model routing with benchmark datasets. *arXiv preprint arXiv:2309.15789*, 2023.

Singh, A. K., Wang, H., Attaluri, L. N. S., Chiam, T., and Zhu, W. Asynchronous verified semantic caching for tiered LLM architectures. *arXiv preprint arXiv:2602.13165*, 2026.

Stojkovic, J., Zhang, C., Goiri, Í., Torrellas, J., and Choukse, E. DynamoLLM: Designing LLM inference clusters for performance and energy efficiency. In *2025 IEEE International Symposium on High Performance Computer Architecture (HPCA)*, pp. 1348–1362, 2025. doi: 10.1109/HPCA61900.2025.00102.

Tam, Z. R., Wu, C.-K., Tsai, Y.-L., Lin, C.-Y., Lee, H.-y., and Chen, Y.-N. Let me speak freely? a study on the impact of format restrictions on large language model performance. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 1218–1236, 2024. doi: 10.18653/v1/2024.emnlp-industry.91.

Tian, K., Mitchell, E., Zhou, A., Sharma, A., Rafailov, R., Yao, H., Finn, C., and Manning, C. D. Just ask for calibration: Strategies for eliciting calibrated confidence scores from language models Fine-Tuned with human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 5433–5442, 2023.

TypeSafe AI. Jev: a System One decision model. `https://docs.typesafe.ai`, 2026. Version 1.13.

Varangot-Reille, C., Bouvard, C., Ciancone, M., Gourru, A., Schaeffer, M., and Jacquenet, F. Doing more with less: A survey on routing strategies for resource optimisation in large language Model-Based systems. *Journal of Artificial Intelligence Research*, 86, 2026. doi: 10.1613/jair.1.19801.

Vendrow, J., Vendrow, E., Beery, S., and Madry, A. Do large language model benchmarks test reliability? *arXiv preprint arXiv:2502.03461*, 2025.

Willard, B. T. and Louf, R. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Xiong, M., Hu, Z., Lu, X., Li, Y., Fu, J., He, J., and Hooi, B. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *International Conference on Learning Representations (ICLR)*, 2024.

Xu, B., Peng, Z., Lei, B., Mukherjee, S., Liu, Y., and Xu, D. ReWOO: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*, 2023.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yang, Z., Adamek, K., and Armour, W. Part-time power measurements: nvidia-smi's lack of attention. *arXiv preprint arXiv:2312.02741*, 2023.

Yao, S., Shinn, N., Razavi, P., and Narasimhan, K. τ-bench: A benchmark for tool-agent-user interaction in real-world domains. *arXiv preprint arXiv:2406.12045*, 2024.

You, J., Chung, J.-W., and Chowdhury, M. Zeus: Understanding and optimizing GPU energy consumption of DNN training. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, 2023.

Zaharia, M., Khattab, O., Chen, L., Davis, J. Q., Miller, H., Potts, C., Zou, J., Carbin, M., Frankle, J., Rao, N., and Ghodsi, A. The shift from models to compound AI systems. BAIR Blog, February 2024. https://bair.berkeley.edu/blog/2024/02/18/compound-ai-systems/.

Zhang, Z., Yao, S., Liu, Z., Zhang, Z., Lai, W. I., Yeung, D. H. H., Zhang, C. K., Ma, F., Yuan, Y., Jiang, Y., and She, D. Similarity is not validity: Defending LLM semantic caches against poisoning. *arXiv preprint arXiv:2609.35908*, 2026.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Zheng, L., Yin, L., Xie, Z., Sun, C., Huang, J., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., Barrett, C., and Sheng, Y. SGLang: Efficient execution of structured language model programs. In *Advances in Neural Information Processing Systems 37 (NeurIPS)*, 2024.

AI USAGE DISCLOSURE

We used generative AI tools extensively in this work, mainly Claude Code (an agentic coding assistant) running Claude Opus 5.5, and Claude Sonnet 5 for a few coding tasks. We describe each use below. The authors directed every step, reviewed all output, and take full responsibility for the code, the results, the citations, and the text.

Ideation and experiment design. The research question, the three decision points, the set of deciders, and the vendor claims to test come from a project plan the authors wrote before any AI assistance. During the work we used the assistant to discuss design alternatives and to check the evaluation protocol for leakage and confounds, for example the tie-breaking and the clustered bootstrap. The authors made all design decisions.

Literature search. The assistant helped find related work and prepare bibliography entries. The authors checked every cited work against its original source: that it exists, that its authors and venue are correct, and that it supports the claim it is cited for.

Code. The assistant wrote most of the code under the authors' specification: the shared decider interface and its adapters, the harnesses for the three decision points, the cluster job scripts, the energy measurement, the analysis and bootstrap scripts, and the scripts that produce every figure and table. The authors reviewed the code and checked its outputs against hand-computed cases and raw run logs. The code is released with the paper.

Running experiments. The assistant helped launch, monitor, and debug jobs on our GPU cluster and diagnosed integration failures with hosted APIs. Every reported number comes from logged runs, which we release.

Data. Claude Opus 5.5 relabeled the query pairs of the two vCache datasets under a guideline we fixed in advance (Section 5; Appendix A). This relabeling is part of the method, and its limits are discussed there. No other data was generated by AI.

Writing. The assistant drafted the text of the paper from the authors' results, outlines, and instructions, and carried out editing passes against a written set of review rules for clarity, consistency of numbers, and the page limit. The authors reviewed and revised every section and every change, and checked each claim against the results.

Models under study. The decision models and LLMs evaluated as deciders (Section 3) are the objects of study. Calling them is part of the experiments, not assistance in producing the paper.

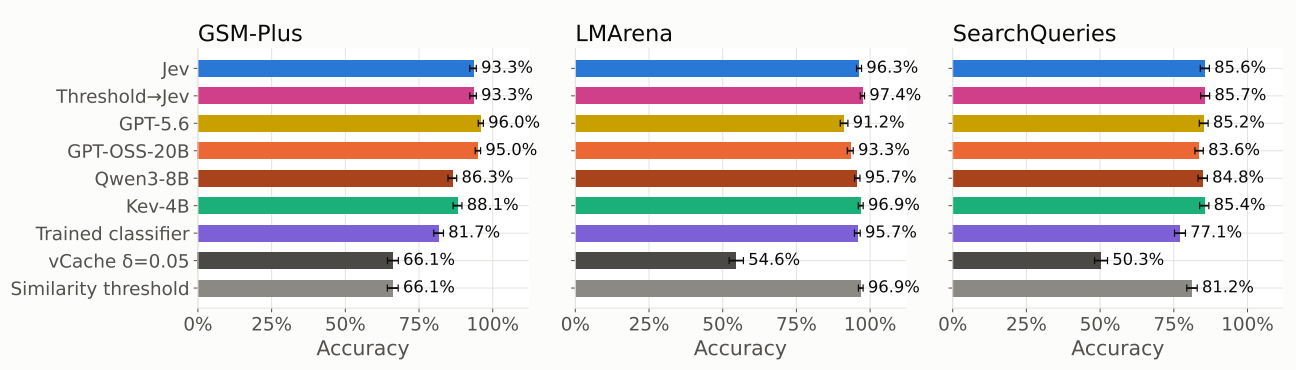


Figure 4. Cache-guard accuracy per decider and dataset with 95% CIs clustered by class. vCache as in Table 4.

A Label Correction for the Two vCache Datasets

On our first run, every LLM decider reached only 70–72% on SearchQueries, and Jev and the frontier model 71% on LMArena, against 93–97% for the hosted deciders on GSM-Plus. Samples of the disagreements put the fault in the labels: "best workout plans for women" and "best workout plan for women" were labeled different, while "count blank cells" and "count non blank cells" were labeled the same.

In SearchQueries, classes are transitive closures of pairwise LLM judgments within embedding clusters. That is valid only if "same" is a strict equivalence, and three things break it. (1) A cached answer is written for its own query: a general query may be judged close enough to two specific ones, but the answer stored for one specific query is wrong for the other. (2) Tolerance accumulates along a chain of individually acceptable links. (3) One mistaken link merges two whole classes, labelling pairs no model ever compared.

In LMArena, a class is "generated from the same seed prompt". An LLM wrote the variants and nothing checked that a variant still asks the same thing; some reverse a premise or change an exam code or a number. In the other direction, identical prompts that came from different seeds are labeled different (37 of the 2,600 pairs).

Correction. We label the pair directly: the new query and the cached query it would be served from. A written guideline, fixed in advance, defines the same request strictly (rewording is the same; any change of entity, number, direction, task, scope or output constraint is different). A frontier LLM (Claude Opus 5.5) judged all 5,200 dev and test pairs blind, seeing neither the original label nor the similarity nor any decider's output. Pairs where a confident verdict disagreed with the original label were judged again, blind, by a separate instance, and a label changed only when both agreed with high confidence. This changed 156 of 2,600 LMArena labels (6.0%) and 409 of 2,600 SearchQueries labels (15.7%). On the corrected labels, the first run's SearchQueries accuracy rose by 10 to 14 points for every LLM decider and for the similarity threshold.

Limits. The corrector is an LLM, and no person checked its labels, so the corrected labels may sit closer to an LLM decider's idea of the same request than human labels would. It judged every pair, not only those where deciders disagreed with the label, so that which pairs were judged did not depend on any decider. GSM-Plus labels are exact numeric answers and were not touched.

B Additional Results

Figure 4 gives the cache-guard accuracies of Table 4 with their confidence intervals, Figures 5–7 add detail to the decision-only replay of Section 4, and Figure 8 to the in-loop arms.

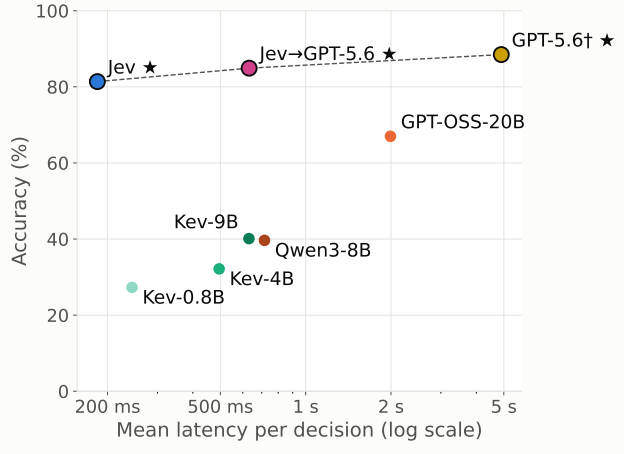


Figure 5. Decision-only replay: accuracy against mean latency per decision (log scale). Stars mark the accuracy–latency frontier; † as in Table 2.

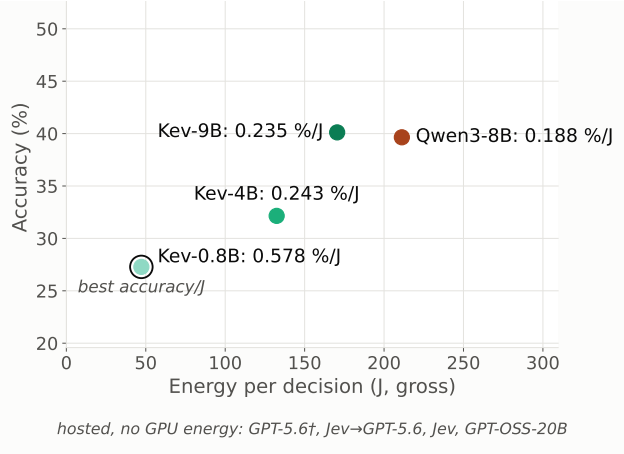


Figure 6. Decision-only replay: accuracy against gross GPU energy per decision for the local deciders on one A100-80GB. Point labels give accuracy points per joule (%/J).

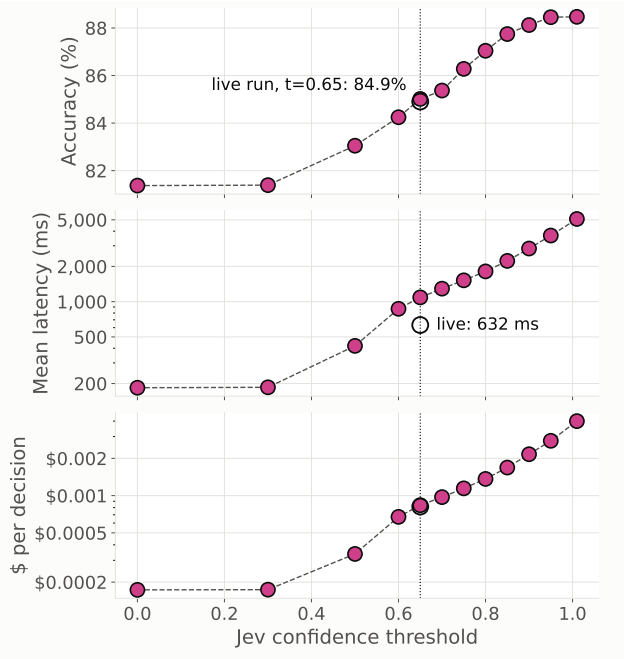


Figure 7. The Jev → GPT-5.6 cascade as its confidence threshold varies (escalate below it). Points are simulated from the standalone replays (latency and cost on log scales), the open ring is the live run at the chosen threshold of 0.65, and the dotted line marks that threshold.

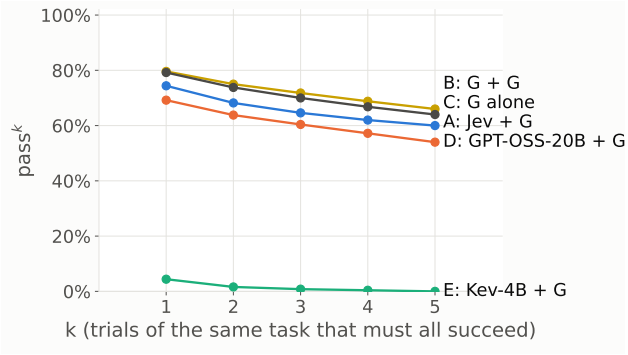


Figure 8. In-loop reliability: pass^k, the chance that k trials of the same task all succeed, per arm.