

A Survey on Jev: Prior Research, Adaptation Costs, and Evidence of System One Models

Evan Chen, Jiamu Zhang, Tianze Yang, Kelly Wan, Liangjie Hong and Liang Wu
Nokia, Sunnyvale, CA, USA

Abstract—Many decisions in LLM systems do not need generated text. They only need a choice among a few fixed options, such as which model should handle a request or whether a response is safe. Typed decision models, also called Jev-style or System One decision models, are designed for exactly this case. They take a state and a question specified at call time, and instead of generating text, they return a probability distribution over a closed answer set. TypeSafe AI offers this interface in Jev, and several open implementations reproduce it. This short survey explains what these models are, where their ideas come from, and what the current evidence can support. These models build on earlier work, which we organize into two lineages. The first lineage, routing research, supplies the goal: a decision component that adapts when the system changes, without retraining. The second lineage supplies the mechanisms: zero-shot classifiers, safety guards, and label-probability readout. How much "no retraining" actually saves, however, depends on what changes. We therefore examine the adaptation cost of three kinds of change: new models, new decisions, and new deployment constraints. A new model is not free, because the routing approaches reviewed here still rely on candidate-specific evidence for each new candidate. A new decision, by contrast, is genuinely cheap, because the shared model needs no retraining. Finally, a new deployment constraint, such as a target error rate, still requires labeled examples. The empirical evidence is also mixed. On the positive side, independent studies report valid outputs and large cost reductions compared with LLM judges, although these savings shrink when the baseline is already cheap. On the negative side, the same studies find that accuracy trails frontier LLMs on hard tasks, that calibration varies across workloads and subgroups, and that added context can change decisions. Given these trade-offs, we recommend testing a typed model against two simpler alternatives before adopting it: a label-probability readout on an open model, and a cheap generative cascade. The comparison should use the same acceptance rule the deployment will use.

Index Terms—Jev, typed decision models, System One models, LLM routing, zero-shot classification, calibration, selective prediction, guardrails.

I. INTRODUCTION

An LLM agent decides which tool to call next and when to stop [2]. A system around it may also choose which model should handle a request [3] and whether a proposed action is risky [4]. Many of these decisions can be posed over a known set of answers. The software that consumes the answer needs a value from that set, ideally with a probability it can compare with a threshold.

Nokia develops AnyJev. E. Chen is an author of CARGO, and J. Zhang, T. Yang and L. Wu are authors of AnyJev. Both systems are discussed in this survey.

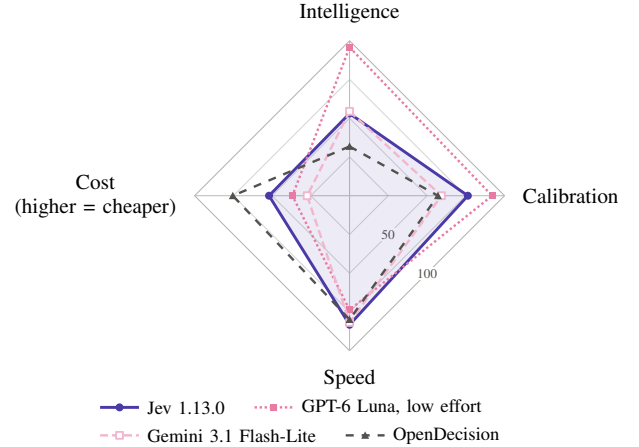


Fig. 1. Scores of four systems on JevBench v1.4.2.2 [1], from three classes: a typed decision model (Jev 1.13.0), two prompted LLMs (GPT-6 Luna at low reasoning effort and Gemini 3.1 Flash-Lite) and a zero-shot ModernBERT classifier (OpenDecision). Each axis is the benchmark's 0–100 score on the same decisions. Intelligence is chance-corrected accuracy across the benchmark's tiers, calibration measures how well stated probabilities match outcomes, speed maps latency to a score (100 at 0.1 s, 80 at 1 s), and cost maps the price per 1,000 decisions to a score, so that higher means cheaper. OpenDecision's speed and cost are the benchmark's estimates rather than measurements. The results file reports no uncertainty estimates. On this snapshot, no system is best on every axis. Against Gemini 3.1 Flash-Lite, the typed decision model scores 1.4 points lower on intelligence and higher on the other three axes. The speed scale is logarithmic: Jev's median latency is 0.65 s, against 1.44 s for GPT-6 Luna at low effort. We chose one system per class to compare the classes, not to rank systems. Three open-weight typed models rank above Jev on the benchmark's composite score.

Asking a generative LLM for such a decision has three drawbacks: 1) the caller pays for decoding and, with reasoning models, for thinking tokens [5]; 2) the LLM returns text that must be parsed; and 3) its reported confidence, if exposed at all, requires validation for the model, task and output format. Token probabilities can be well calibrated in some formats [6]. The classical alternative is a conventional classifier, trained for a fixed task and label set. It is fast and often accurate, but changing the task or labels often requires adapting the classifier or its output head.

Typed decision models sit between these options. They take a state and a question whose type, wording and answer set are supplied at call time, and they return a probability distribution over the closed answer set. TypeSafe AI offers the interface in Jev, which it describes as a System One model, after Kahneman's fast, intuitive System 1 [7], [8]. We use the term *typed decision model* for these models, which are also called Jev-style or System One decision models. Several open

implementations now expose the same interface [9]–[14], and OpenAI has announced a related Decisions API in limited preview [15]. On one benchmark snapshot, none of the four systems in Fig. 1 is best on every axis [1].

Typed decision interfaces package established work on classification, probability readout, calibration and selective prediction into caller-defined choices, scores and Boolean judgments for repeated use in software. The closest prior work is Tang and Zheng’s audit of the 28 arXiv papers on these models posted between September 19 and 24, 2026 [16]. Their audit traces the interface to label-probability classification, constrained decoding, reranking, calibration and cascades. Two concurrent working drafts also review the area: one traces a lineage from classical classifiers to Jev, including routing and cascades [17], and the other reviews calibration, selective control and open implementations [18]. Building on these accounts, this short survey makes three contributions:

- 1) **Two lineages.** We organize prior work into two lineages. Routing supplies the goal of a decision component that adapts to change without retraining, and classification and label readout supply the mechanisms (Section III).
- 2) **Adaptation cost.** We describe what each change requires, for new models, new decisions and new constraints, and we separate newly acquired resources from reused ones (Section IV).
- 3) **Evidence and guidance.** We summarize what independent studies support, the main deployment risks, and a pilot protocol with starting comparisons for practitioners (Sections V–VII).

Scope and sources. This survey covers public work through September 30, 2026. We searched arXiv, GitHub and official documentation. Each search combined a name for Jev or its interface (Jev, System One model, typed decision model, decision model) with a term for one surveyed family (LLM routing, cascades, zero-shot classification, policy-conditioned guards, rubric judges, label-token readout, calibration). We then followed citations from the works found. Most work specific to typed decision models appeared in the preceding two weeks as preprints or repositories that have not been peer reviewed. Jev has no technical report, so our description of it follows TypeSafe’s documentation [8]. We call a study *independent* when its authors neither developed nor funded the system it measures, and we attribute every result to the researchers who measured it. We distinguish documented interface properties from empirical findings and identify results supported by a single study. Related surveys cover routing [3], [19] and LLM judges [20].

II. TYPED DECISION MODELS

A. The Interface

We describe the interface through Jev 1.13, following TypeSafe’s public documentation [8].

Calls and question types. A call carries a *state*, which is text, a JSON object or an array of text values. It also carries a set of keyed questions of three types: 1) a *choice* question selects one option from a caller-defined set of at most 255 options, and the answer gives a probability for every option;

2) a *score* question places the state on an ordered rubric of 2 to 10 levels, and the answer gives a probability for every level and the probability-weighted level; and 3) a *Boolean* question, which TypeSafe calls *noul*, asks whether a statement is true, and the answer is the probability of yes. Types can be mixed in one call. The questions of a call are evaluated against the state independently of one another.

Pricing and versions. The service is priced per input token, and output tokens are free. The same weights serve every account, so a caller adapts Jev only through the state and the text of its questions. Model aliases resolve to versioned identifiers, and the documentation advises pinning a version whenever thresholds have been tuned. Because the service is hosted, every state is sent to TypeSafe. When the state must stay on the team’s own infrastructure, only the open-weight rows of Table I apply.

Training. TypeSafe has published no technical report, weights, parameter count, architecture or description of its training data. Its documentation states that customer requests are not used for training. The one named training method is reinforcement learning for calibrated decisions (RLCD). TypeSafe describes RLCD as training the model to return decisions and calibrated probabilities instead of generated text.

Documented limitations. Jev reads instructions literally and is unreliable on counting, arithmetic, date comparisons and questions that need several hops of reasoning. Its accuracy falls when the state contains unrelated material. It does not treat the state as hostile, so text in the state can steer its answer. A Boolean question and its negation need not receive complementary probabilities. TypeSafe lists these limitations for *jev-1.13* and pairs each with a workaround, such as keeping arithmetic in code.

A worked call. Fig. 2 shows one call with illustrative values. The state is an agent’s proposed tool call together with the user’s request. The caller asks one question of each type, and the model returns one distribution per question. The caller’s code then applies a threshold to each answer and combines the results. The thresholds, and the decision to act, abstain or escalate, belong to the caller rather than to the model.

B. The Typed Decision Contract

We judge a typed decision model by four properties: type safety (P1), calibration (P2), invariance (P3) and cost (P4).

Type safety (P1). Every answer lies in the declared answer set. This is structural validity, not correctness.

Calibration (P2). For a *choice* question, predictions assigned a given top probability should be correct at that rate. The same condition applies to the probability of yes for Boolean questions and to each cumulative event for *score* questions. Calibration is measured on a deployment distribution, so a model calibrated for one question and one traffic mix need not be calibrated for another, or within every subgroup. The studies in Section V report the expected calibration error (ECE), a binned estimate of this gap.

Invariance (P3). Reordering the options should not change the probability each option receives. Renaming the options while their definitions are held fixed should not change it either.

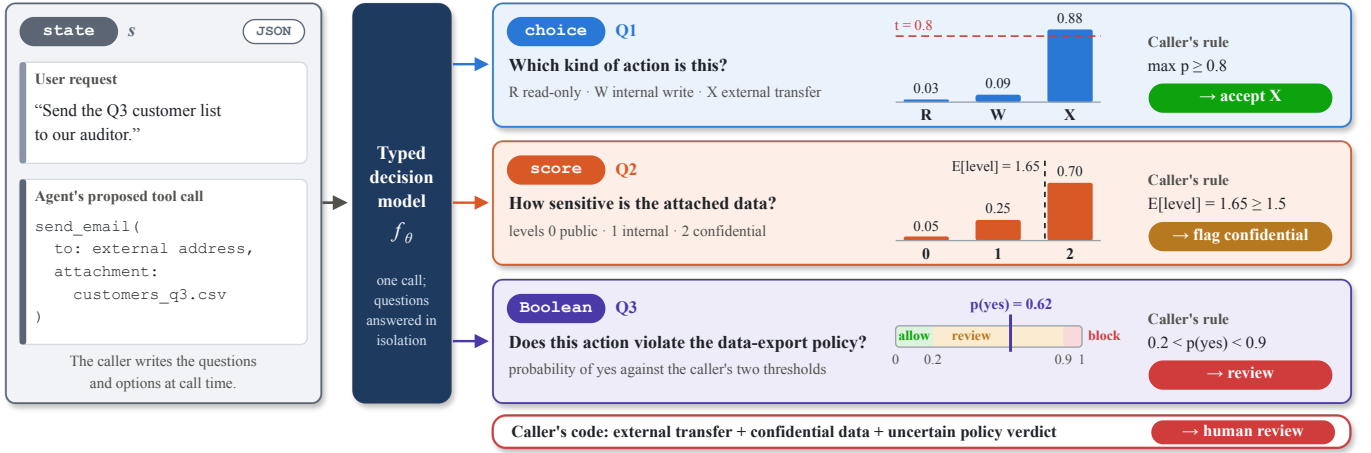


Fig. 2. A worked call to a typed decision model, with illustrative rather than measured values. One state is evaluated against three questions, one of each type. The model returns a distribution over each question’s closed answer set: bars show the *choice* and *score* distributions, and the Boolean answer is the probability of yes, shown against the caller’s allow and block thresholds (shaded band: review). The caller, not the model, sets the thresholds and the rule that combines them.

P3 also covers coherence: a question and its negation should receive complementary probabilities.

Cost (P4). Cost covers forward passes, network round trips, latency and price per decision. Reports should state what the latency includes, for example model time only or the full round trip.

Type safety is an interface requirement. Calibration, invariance and cost describe behavior that must be measured rather than assumed. Constrained decoding also guarantees type safety [21], but the model still generates tokens and exposes probabilities only where the provider returns token log-probabilities. LLM judges accept a rubric at call time [22], [23], but they usually report a score or verdict, often with a rationale, rather than a calibrated probability [20].

Software acts on a decision through an *acceptance rule* that the caller sets. A *choice* is typically executed when its top probability exceeds a threshold and escalated otherwise. A Boolean gate that asks whether an action is unsafe can use two thresholds. It allows the action below the lower one, blocks it above the upper one, and asks for human review in between. When low-confidence decisions are deferred in this way, the model acts as a selective predictor [24]. We call the share of decisions resolved without human review or escalation the *automation coverage*. For a Boolean gate, it includes both automatic allows and automatic blocks.

C. Implementations

Beyond Jev, open implementations fall into three families (Table I): 1) trained encoder heads, such as Laya, which fine-tunes ModernBERT-class encoders [9], [25]; 2) fine-tuned decoders, such as Kev, Nimble and Decider, which adapt open decoder models [10]–[12]; and 3) readout from frozen open models, such as SemIf and AnyJev [13], [14]. Laya, Kev, Nimble and Decider expose *TypeSafe*-compatible endpoints [9]–[12], so a team can send identical requests to them and to Jev. Compatibility does not establish equivalent decision behavior.

Most calibration claims for the open implementations also come from their own developers.

III. TWO LINEAGES: ROUTING AND READOUT

We connect typed decision models to two lines of work: routing frames the problem of adapting a decision component to change, while classification and readout provide mechanisms for answering questions specified in the input. Routing is also only one of their applications: Tang and Zheng identify one routing application among the 28 papers they audit [16]. This section traces both lineages.

A. The Problem Lineage: Routing

Classical routers allocate queries among models, including through cascades [26], [27]. Each router answers one question fixed at design time, usually over a pool of models fixed at training time. Routing surveys list adding candidates without retraining as an open goal [3] and place it within the broader challenge of retraining-free transfer [19]. Recent routers relax the fixed design in three ways:

- 1) **Profile routers** separate model information from shared router parameters. A new model enters through a profile, such as a per-cluster error vector in UniRoute [28] or neighbor accuracies in a tuned k -nearest-neighbor (kNN) router, which matches and often outperforms learned routers [29]. Building a profile still requires the new model’s graded outputs on a reference set. Descriptor routers move part of the profile into text: GraphRouter initializes a new model from its description but still uses few-shot interaction data [30], and Router-R1 reads descriptors of price, latency and example performance [31].
- 2) **Route-as-text routers** read route descriptions at call time. Arch-Router receives route policies in its prompt and returns the chosen policy, while a human-maintained table maps policies to models [32]. Its authors report that overlapping policy descriptions degrade routing.

TABLE I

IMPLEMENTATIONS OF THE TYPED DECISION INTERFACE, GROUPED BY MECHANISM FAMILY. “USER LABELS” COUNTS THE LABELED EXAMPLES A USER SUPPLIES PER NEW QUESTION TO OBTAIN PROBABILITIES; DEVELOPER TRAINING DATA ARE EXCLUDED. EVERY ROW STILL NEEDS LABELED EVIDENCE TO SET AN ACCEPTANCE RULE FOR A RISK TARGET (SECTION IV). “PASSES” COUNTS MODEL PASSES FOR ONE QUESTION. L0, L1 AND L2 ARE ANYJEV’S ZERO-LABEL, TEMPERATURE-FITTED AND CLOSED-FORM-HEAD LEVELS, WITH LABEL COUNTS AS STATED BY ITS DEVELOPERS. K : NUMBER OF OPTIONS.

System	Mechanism and training	User labels	Passes
HOSTED, CLOSED WEIGHTS			
Jev 1.13 [8]	Vendor-trained with RLCD; unpublished	0	One API call, which several questions can share
OPEN WEIGHTS: TRAINED ENCODER HEADS			
Laya [9]	Fine-tuned ModernBERT or mmBERT	0; developers advise task fine-tuning and local temperature fitting	One encoder forward pass
OPEN WEIGHTS: FINE-TUNED DECODERS			
Kev [10]	Pointer head on Qwen3.5 or Qwen3.8; LoRA, full fine-tuning at 27B	0	One prefill
Nimble [11]	LoRA on Qwen3.5-9B	0	One prefill
Decider [12]	Fine-tuned Qwen3.5; also a Gemma-4 LoRA and stock-model readout variants	0	One prefill
OPEN WEIGHTS: READOUT FROM FROZEN MODELS			
SemIf [13]	Logit readout from frozen checkpoints	0	One prefill
Any-Jev [14]	Debiased readout (L0); temperature (L1); closed-form head, no gradient training (L2)	0 (L0); 100–500 (L1); 100–300 (L2)	Up to K prompts sharing the state prefix (L0, L1); one partial forward pass (L2)

- 3) **Threshold control** meets a deployment constraint by moving a threshold on a score. RouteLLM varies its cost–quality trade-off this way [27]. Agreement pivots escalate a query to a stronger model when several answers from a cheap model disagree with one another. Yue et al. measure agreement across chain-of-thought and program-of-thought answers [33]. CARGO measures it across equivalent system prompts and calibrates the pivot of a probabilistic offloading rule on an unlabeled warmup batch to meet a target *offload ratio*, the share of queries sent to the stronger model [34].

LLMRouterBench evaluates fixed pools, so it does not test new-model generality directly. On those pools, several recent routers do not reliably outperform a simple baseline, and the remaining gap to the oracle comes mainly from failures to recall the right model [35]. UniRoute separately evaluates routing to models held out from router training, using profiles computed on a validation set [28].

Typed decision models as routers. Three studies run Jev as

an LLM router. In all three, the knowledge about candidate models comes from outcome data outside the decision model:

- 1) On a 13-model pool from LLMRouterBench, TokenTrim combines Jev’s rating of query difficulty with nearest-neighbor accuracies of each candidate. Its ablation without Jev reaches the same quality–cost frontier, and the authors attribute the gain to the retrieved evidence [36].
- 2) The jev-router study trains a router on Jev’s answers to six typed questions. In a pre-registered replication on fresh items, routers without Jev features matched or beat it at two of three quality floors [37].
- 3) SeLMRoute trains a supervised performance model on Jev’s answers to sixteen typed questions. In grouped out-of-fold evaluation on LLMRouterBench, it reports 72.64% accuracy against 69.23% for the strongest fixed candidate, without establishing monetary savings [38].

In these studies, typed questions provide features about the query, while candidate performance is estimated from outcome data. Adding a candidate therefore requires new candidate-specific evidence, supplied in the state or used by the surrounding router.

B. The Mechanism Lineage: Classification and Readout

The open implementations descend from four families in which a fixed model answers a decision specified in its input. **Label-token readout.** A language model can answer a closed question by reading the probability it assigns to each candidate answer instead of generating one. Kadavath et al. introduced $P(\text{True})$, the probability a model assigns to “True” when asked whether a proposed answer is correct [6]. Such readouts carry two documented biases: 1) a prior toward labels that are frequent or recent in the prompt or common in pretraining data, which contextual calibration and Batch Calibration reduce [39], [40]; and 2) a preference for particular option identifiers or positions, which PriDe estimates and removes without labels [41]. For a *choice* question, the second bias means that reordering the options can change the decision, which violates invariance (P3).

Zero-shot label-as-input classifiers. Yin et al. cast zero-shot text classification as entailment, scoring each candidate label as a hypothesis about the text [42]. GLiClass encodes all labels together with the text in one pass [43]. The encoder implementations in Table I descend from this line.

Policy-conditioned guards. Llama Guard places adjustable taxonomy categories in its prompt and uses the probability of the token `unsafe` as its score [44]. DynaGuard takes free-form rules and offers a fast mode that emits a pass or fail verdict directly [45]. Guards answer one Boolean question against a policy, sometimes adding the violated categories or an explanation. The fine-tuned decoder implementations generalize these single-verdict guards to arbitrary typed questions.

Rubric judges. Prometheus 2 evaluates responses under user-defined criteria [22]. G-Eval computes a score as the probability-weighted sum over score tokens [23], which is the readout of the `score` type. Judge reliability is usually measured as agreement with human ratings rather than as the calibration of a probability [20].

What the families share. Across these families, the caller specifies labels, policies or rubrics in the input. Typed decision interfaces expose such decisions through `choice`, `score` and Boolean questions, while implementations differ in their training and readout mechanisms. Tang and Zheng find no demonstrated accuracy advantage from typed readout alone over comparable label-probability readouts [16].

IV. ADAPTATION COST: WHAT EACH CHANGE REQUIRES

A deployed decision component serves a pool of candidate models, a set of decisions and a set of deployment constraints, such as an offload ratio, a token budget, a latency target or a risk target. Each of these can change after deployment, which gives three axes of change: 1) new models (**M**), 2) new decisions (**D**) and 3) new constraints (**C**).

Reusing the model is not free. Many changes leave the shared model’s weights unchanged. They may still require per-change state, such as a model profile, question text, a threshold or a small head fitted for one decision. Setting this per-change state has an *adaptation cost* with four parts: 1) outcome labels, 2) calls to profile a model, 3) unlabeled warmup samples and 4) the specification text a person must write. We count newly acquired resources separately from reused ones. For example, reusing an existing calibration set costs no new labels but still relies on labeled evidence.

A sequence of changes. Consider a support service that classifies tickets and routes difficult cases to an LLM. Adding a ticket category is a new decision: the team writes its definition and checks it on a small set of labeled tickets, while the shared model stays frozen. Adding an LLM is a new model: the team records its outputs on a reference set and supplies the resulting profile. Lowering the offload ratio is a new constraint: the team picks a new threshold from representative warmup scores. Tightening the permitted error rate is also a new constraint, but it needs a risk-control procedure. The team may reuse its existing labeled set if the traffic, the question text and the model version have not changed since the set was labeled.

What each family needs. Table II lists what a change on each axis requires, family by family. Typed decision models accept a new decision as question text, without changing the shared model. Testing the new decision still takes labeled examples. The other two axes need further information. In the routing approaches reviewed here, a new candidate model needs candidate-specific evidence, such as a profile (Section III-A). An offload ratio needs representative unlabeled scores. A risk target needs a validated acceptance rule and labeled evidence.

Guarantees need labeled evidence. The methods that set or certify an acceptance rule use labeled data. Temperature scaling fits one parameter on labeled data to reduce over-confidence [46], but it carries no finite-sample guarantee. Selective prediction chooses a confidence threshold on held-out labels so that the accepted decisions meet a risk level with high probability [24]. Conformal prediction turns a labeled calibration set into prediction sets with a coverage guarantee under exchangeability [47]. Conformal risk control extends the guarantee to a bounded monotone loss, such as a miss rate [48]. Trust or Escalate applies a labeled threshold of this

kind to a cascade of LLM judges, with a guarantee of human agreement [49]. These guarantees hold on the distribution they were calibrated on. They need not hold within every subgroup or after the model or the population changes, and calibration is known to degrade under dataset shift [50]. A changed question, population or model version can therefore require new labeled evidence.

V. WHAT THE EVIDENCE SHOWS

Most studies of typed decision models are preprints or repositories from the two weeks before our coverage date.

A. *Outputs Are Valid, but Validity Is Not Correctness*

Independent evaluations report no schema violations in their tested calls. A calibration repository recorded zero schema failures in 4,621 calls [51]. Sun et al. observed no type errors even in settings where accuracy collapsed [52]. These results support structural validity under the tested conditions. They also show that a valid answer can be wrong at high confidence, as several of the studies below record.

B. *Cost and Latency Depend on the Comparator*

The largest savings appear against generative baselines that decode a verdict for each decision. Rao et al. find that LLM judges cost 16 to 325 times as much as Jev in their paper’s second revised version [53]. Ibrahim et al. measure a median cost 44 times lower than that of the best LLM annotator for each task, although their cheapest LLM baseline costs only about 1.6 times as much as Jev [54]. The ratio shrinks when the baseline is cheaper. Against LLM-judge scorers for alignment-failure detection, Guo et al. find Jev 63 times cheaper at list prices. When every judge is repriced at GPT-4o-mini rates and Jev answers one generic question, Jev is 12 times cheaper pooled and 3.3 times cheaper at the median benchmark [55]. Inside an agent loop, the comparison depends on the benchmark. On the REFLEX authors’ own benchmark, their typed cascade succeeds more often than a cheap generative cascade and calls the strong model less often. On 60 paired τ^2 -bench tasks, it costs 3.7 times less than a strong-only agent. There, however, the cheap generative cascade costs \$0.041 per episode against \$0.057 and is not less accurate [56].

Part of the speed-up comes from not generating text. SemIf’s developer reports 21 decisions in 1.023 s, against 5.332 s when the same decisions are generated as a compact JSON array [13]. In a vendor example with Jev in both arms, asking 13 questions about one document in a single call took 0.27 s, against 2.71 s summed over 13 sequential single-question calls on `jev-1.12` [8].

C. *Accuracy on General and Hard Tasks*

On standard classification and language-understanding benchmarks, an evaluation of 346,009 requests over 37 datasets with a pinned model version reports 95–99% accuracy

TABLE II

WHAT A CHANGE ON EACH AXIS REQUIRES, BY FAMILY. **M**: A NEW CANDIDATE MODEL; **D**: A NEW DECISION; **C**: A NEW DEPLOYMENT CONSTRAINT, EITHER AN OFFLOAD RATIO OR A RISK TARGET.

Family (examples)	M New model	D New decision	C New constraint
ROUTERS AND GATES			
Profile routers (UniRoute [28]; kNN [29])	A profile built from the model’s graded outputs on a reference set	Not supported: one fixed routing question	A cost weight applied to predicted per-model errors; labeled evidence for a risk target
Route-as-text routers (Arch-Router [32])	An edit to the human-maintained policy-to-model table	A new route description; routes only	Not addressed
Agreement pivots (Yue et al. [33]; CARGO [34])	A capable cheap model with comparable answers, and a retuned agreement threshold	One fixed accept-or-escalate question	A threshold on agreement (CARGO recalibrates it on an unlabeled warmup batch per offload ratio); labeled evidence for a risk target
DECISIONS SPECIFIED IN THE INPUT			
Zero-shot classifiers (NLI [42]; GLiClass [43])	Not used for model selection	New label strings	A threshold on label scores; labeled evidence for a risk target
Policy-conditioned guards (Llama Guard [44]; DynaGuard [45])	Not used for model selection	New policy text; Boolean verdicts, plus the violated categories in Llama Guard	A threshold on the probability of an unsafe or failing verdict; labeled evidence for a risk target
Rubric judges (Prometheus 2 [22]; G-Eval [23])	Not a router; can grade a new model’s outputs to build its profile	A new rubric; scores and pairwise preferences	A threshold on scores; labeled evidence for a risk target
TYPED DECISION MODELS			
Jev and open implementations (Table I)	Candidate-specific evidence, such as a profile built from graded outputs, supplied in the state or used by the surrounding router	New question text, and labeled examples to test it	Warmup scores for an offload ratio; a validated acceptance rule with labeled evidence for a risk target

on IMDB, SST-2, HellaSwag and ARC [57]. In the same evaluation, Jev scores higher than next-token readouts of Qwen3.8-27B on 27 of 37 datasets and higher than those of Gemma-4-E4B on all 37. It reports 57.9% on five-way SST-5, although its SST-5 scores correlate with human ratings at 0.851 when treated as an ordinal scale. The gap to frontier LLMs opens on hard, subjective and specialist tasks. On computational-social-science annotation, Jev trails the best LLM for each task on 14 of 15 tasks. The authors note that choosing the best LLM on the test items favors the LLMs [54]. Against LLM judges on rubric criteria, Jev’s accuracy differs significantly in at most 8 of 27 paired comparisons [53]. As a rubric judge, Jev comes within three points of GPT-6 where a verdict can be read off the text, but falls behind where it must be derived, as in math, code and logic, in the paper’s third version [58]. On the hard tier of JevBench v1.4.2.2, 220 decisions that include long policy documents, ambiguous cases and multi-hop lookups, Jev answers 74.1% correctly, against 97.3% for GPT-6 Luna at low reasoning effort and 75.0% for Gemini 3.1 Flash-Lite [1]. Claude Opus 5 and GPT-5.6 Sol wrote these items and checked each other’s gold answers, which may favor frontier LLMs. Date and number items account for 22 of Jev’s 57 errors there, a weakness TypeSafe documents [8]. On the same snapshot, the benchmark’s composite score, a harmonic mean of the four axes in Fig. 1 with penalties for low axis scores, ranks Jev fourth of 91 ranked systems, behind three open-weight typed models. Tang and Zheng add two qualifications [16]: 1) reported parity often comes from tests on which strong models are near ceiling, which does not establish equivalence; and 2) typed readout has shown no independent accuracy advantage in five studies with label-probability readouts on comparable backbones.

D. Pooled and Per-Workload Calibration

Pooled over public benchmarks, Jev’s choice probabilities are well calibrated. Deußer et al. report an ECE of 0.028 over 279,925 answers on 22 datasets [57]. The same data show the spread that pooling hides: 8 of the 22 datasets exceed an ECE of 0.05, and 20 of the 22 show overconfidence. Guo et al. find a pooled ECE of 0.047 over 31 binary benchmarks, but a median per-benchmark ECE of 0.168, against 0.074 expected for a perfectly calibrated model at these sample sizes [55]. Failures also appear within subgroups. In an agent-security evaluation, 9.20% of the inputs that Jev scored below 0.1 were labeled unsafe, against a mean prediction of 0.205%, in the paper’s second version [4]. Almost all of these unsafe inputs are attack-derived texts without explicit instructions, which the benchmark labels unsafe. Without them, the observed rate is 0.228% against a predicted 0.176%, and Liu treats the comparison as exploratory. On one annotation task, Ibrahim et al. observe high confidence at near-chance accuracy, although Jev’s probabilities are better calibrated than the verbalized confidence of 16 of 19 LLMs [54].

Confidence-based selection also improves accuracy in some workloads. On Banking77, Deußer et al. report 79.7% accuracy on all queries and 96.3% on the half on which Jev is most confident [57]. Tang and Zheng conclude that calibration must be judged per model and per workload [16].

E. Sensitivity to Added Context and Option Names

Several studies change one part of a request and measure how the decision moves. They report three effects:

- 1) **Option names.** Sun et al. swap which definition is bound to the option names yes and no, and keep

everything else fixed [52]. The ranking of the open implementation Laya then reverses: its AUC falls from 0.938 to 0.232. Jev’s AUC falls from 0.81 to 0.58. In the binary setting, replacing the names with random strings largely removed the effect, with little change in accuracy.

- 2) **Added opinions.** In JevAdvBench, one unverified opinion appended to the state flips 12.1% of Jev’s decisions, which is statistically tied with the strongest injected command at 10.1% [59].
- 3) **Optimized context.** JevOut optimizes a fluent addition to the context for a chosen wrong answer and redirects 61.4% of initially correct decisions [60].

Detection can also fail outright. On the WAINjectBench test split, Jev misses all 130 unsafe inputs that carry no explicit instruction in the paper’s second version, and Nimble, the top-ranked model on that benchmark, misses 92.3% of them [4]. Not every attack succeeds, however. Explicit prompt injection shifts Jev’s action probabilities but rarely selects the attacker’s target, even with adaptive attacks [61]. Exploratory analyses in both studies link successful attacks to a small decision margin, the gap between the chosen answer and the alternatives [59], [61]. Coherence is also imperfect. Li et al. find that on 480 negation pairs, Jev’s probabilities for a statement and for its negation miss summing to one by 0.064 on average [62].

VI. DEPLOYMENT RISKS AND MITIGATIONS

The findings above and the interface documentation point to four deployment risks: 1) text in the state can persuade the model, whether it arrives as an instruction or as evidence; 2) whoever names the options, and possibly whoever orders them, controls part of the decision; 3) a silent model update may change the probability scale on which thresholds were tuned; and 4) schema validity or pooled calibration can be mistaken for correctness on a particular workload. Few defenses have been evaluated against these attacks on typed decision models. Most of the steps below are therefore design principles with support from adjacent settings.

Keep fetched content out of the gate (risk 1). Restricting the gate to the original authorization, validated arguments and known policy reduces its direct exposure to untrusted tool output and retrieved content. This is the decision-layer form of separating control flow from untrusted data, the principle behind CaMeL [63]. The authors of JevAdvBench likewise recommend treating the state as untrusted input [59]. The cost is utility, because a decision that legitimately depends on tool output needs a separate, narrower question.

Allow, review or block (risks 1 and 4). We recommend three outcomes for a probabilistic gate, allow, review and block, with human approval for consequential actions. TypeSafe’s documentation likewise divides confidence into three ranges with different actions [8]. On R-Judge at a 1% miss budget, Liu reports that separate allow and block thresholds raised Jev’s automation coverage from 7.63% to 47.03%, mainly through more blocks [4]. Liu also finds that thresholds that pass validation need not hold on test data.

Fit thresholds per question (risk 4). Exploratory analyses link successful attacks to small decision margins. We

recommend fitting each threshold on labeled data from the deployment and adding a review band around it. Section VII-B gives a rule of thumb for the number of labels.

Fix option names and order yourself (risk 2). In binary questions, neutral option identifiers largely removed the re-naming effect [52]. With three or more options, even neutral letters changed about half of Laya’s decisions. Sun et al. did not run this test on Jev. The deployer, rather than any party that registers options, should fix the option order. Where the model is open, the readout can be averaged over option permutations [41]. LLM judges can favor outputs labeled with well-known model names [20], so candidate names should be anonymized when the options are models.

Pin the model version (risk 3). Thresholds are tuned to one model’s probability scale, so a silent update can act as a silent policy change. TypeSafe’s `jev-latest` alias resolves to the most recent stable release, and its documentation advises pinning an explicit version [8].

VII. PRACTICAL GUIDANCE AND OPEN QUESTIONS

A. Where to Start

Table III lists starting comparisons by deployment need. In each case, adoption depends on quality under the acceptance rule the deployment will use, and on total cost, including adaptation, serving and fallback.

Include a readout on an open model. Tang and Zheng find no independent accuracy advantage for typed readout over label-probability readouts on comparable backbones [16]. This does not mean that a team’s own open model matches Jev. In one large evaluation, Jev scored higher than next-token readouts of two open models on most datasets (Section V). We therefore recommend such a readout as a baseline to measure, not as a default. SemIf and AnyJev are open examples, and three of us develop AnyJev.

Check that the fallback corrects errors. Escalation improves accuracy when the fallback corrects enough deferred errors to offset the errors it introduces. Rao et al. find that on Jev’s most confident errors, about 96% of LLM-judge verdicts repeat the wrong answer. As a result, no cascade improves on the best single judge by more than 2.7 points in their paper’s second version, even with oracle thresholds [53]. In contrast, JEV-as-a-Judge accepts Jev’s confident verdicts and escalates the rest to a reasoning judge with a threshold fixed in advance. On 1,610 held-out pairs, this cascade is 0.9 points more accurate than GPT-6 at 41% of its fee, although its confidence routing weakens on style-adversarial pairs [58]. Ibrahim et al. find that escalating low-confidence annotations to an LLM matches the LLM alone at a quarter to half of its cost [54]. A typed first stage should also be compared with a cheap generative cascade, which was cheaper on τ^2 -bench in the one agent study that tested both (Section V).

B. A Pilot in Five Steps

We recommend five steps before adoption: 1) pin a model version; 2) label a sample of the deployment’s own traffic for each question, and split it into a tuning half and a test half; 3)

TABLE III

SUGGESTED STARTING COMPARISONS FOR CHOOSING A DECISION MODEL. EACH ROW GIVES ALTERNATIVES TO TEST AND A CRITERION FOR ADOPTING A TYPED DECISION MODEL, MEASURED UNDER MATCHED DATA AND SERVING CONDITIONS WITH THE PILOT IN SECTION VII-B.

Deployment need	Starting alternatives	Adopt a typed model if
Stable repeated task; labels available	Small fine-tuned encoder or SetFit [64], at a matched label budget	It reaches the required error rate at lower total cost or with less adaptation work
Changing short label sets; few labels	NLI [42], GLiClass [43], and a label-probability readout on an open model, with or without debiasing	It gives better accuracy at the same latency, after counting the passes that many options and debiasing need
Many judgments over one state	One batched LLM call with structured output	Its savings remain when both sides batch, cache and measure latency the same way
Policy screening	A guard trained on a fixed taxonomy, such as Llama Guard [44], and a guard for user-written rules, such as DynaGuard [45]	Its miss rate meets the target on held-out local data, with uncertainty assessed in each prespecified risk group, and it provides higher automation coverage
Changing LLM pool	Tuned kNN [29] or UniRoute [28], given the same outcome data on the candidates	It improves quality or cost beyond what the profiles alone provide, after charging its own calls
Multi-step or specialist judgment	Fast generative model, reasoning reference and a cheap generative cascade	It is accurate enough on the hard cases, or its fallback corrects the deferred errors at acceptable total cost

run the typed model and the alternatives in Table III on identical requests, with the same batching and latency boundary; 4) set the acceptance rule on the tuning half, and report accuracy, automation coverage and error rate on the test half, per relevant group; and 5) with the acceptance rule fixed, repeat the test measurements after reordering or renaming options, negating a Boolean question, and appending an unrelated opinion to the state. The number of test labels follows from the risk target. With the acceptance rule fixed before testing, zero errors among n independent accepted test decisions give an approximate one-sided 95% upper confidence bound of $3/n$ on the error rate [65]. About 300 such decisions are needed to place this bound near 1%. This count covers accepted test decisions, not the whole labeling budget, and a pooled bound does not hold simultaneously for every group. Steps 2 to 5 should be repeated after any change of model version, question or traffic.

C. What to Report

Building on Tang and Zheng’s checklist [16] and our adaptation-cost account, we ask evaluations of typed decision models to report six items: 1) the model version, not only its alias; 2) whether reference labels are gold labels or a teacher model’s labels, since agreement with a teacher is not accuracy; 3) calibration per workload and per relevant group, not only pooled; 4) the latency boundary, and whether the baseline batches its requests; 5) matched baselines, including a small fine-tuned encoder or SetFit at the same label budget [64], a label-probability readout on an open model, and a cheap generative cascade; and 6) newly acquired labels separately from reused ones.

D. Open Questions

A new question can be specified without retraining, but its acceptance rule still needs an evidence base. For a new caller-written question, under what assumptions can labeled evidence from earlier questions support a useful acceptance rule, and how many new labels are needed when it cannot? Conformal

methods, including weighted variants for covariate shift, answer parts of this question under their own assumptions [47]. We propose two studies that would make progress on it:

- 1) **Value beyond profiles.** On one replayable routing benchmark, such as LLMRouterBench [35], compare a tuned kNN router, a text-feature router and a router on typed features. Give all three the same model profiles, and charge the decision-layer calls to the cost. Running the candidates under both real and anonymized names separates prior knowledge of model names from use of the profiles.
- 2) **Transfer across questions.** Hold out whole families of questions, and compare local calibration with reuse of labeled evidence from related questions at stated label budgets. Report automation coverage at a fixed risk target on disjoint test data.

The first study would test whether typed features add predictive value beyond the profiles under matched conditions.

VIII. FREQUENTLY ASKED QUESTIONS

What is Jev? Jev is TypeSafe AI’s hosted typed decision model, which TypeSafe calls a System One model. It takes a state and typed questions defined at call time, and it returns a probability distribution over each question’s closed answer set instead of text (Section II-A).

Are there open-source alternatives to Jev? Yes. Examples come from three families: 1) Laya, a trained encoder; 2) Kev, Nimble and Decider, which are fine-tuned decoders; and 3) SemIf and AnyJev, which read typed decisions from frozen open models (Table I).

Is Jev cheaper and faster than an LLM? Reported savings depend strongly on the baseline and on pricing assumptions. The largest reductions are against LLM judges and annotators, while comparisons with cheap LLMs show much smaller gaps (Section V). Part of the speed-up comes from not generating text. In one agent study, a cheap generative cascade cost less than a typed cascade.

How accurate and well calibrated is Jev? One large evaluation reports 95–99% accuracy on standard classification and

multiple-choice benchmarks, and Jev trails frontier LLMs on hard tasks. Its pooled calibration error is low, but 8 of the 22 datasets in the same evaluation exceed an ECE of 0.05 (Section V).

Is Jev a router? Not in the narrow sense of a model trained to choose among a fixed pool. It shares routing’s goal of a decision component that adapts without retraining, and a caller can pose model selection to it as a typed question. In the router studies reviewed here, knowledge about the candidate models came from outcome data rather than from Jev itself. In two of the three studies that ran Jev as a router, routers without Jev’s answers matched it on held-out data, in one study only in a pre-registered replication on fresh items. The third reports higher accuracy than the best single model but did not establish monetary savings (Section III-A).

Does type safety mean the answers are correct? No. Type safety guarantees that every answer lies in the answer set. Independent studies record valid answers that are wrong at high confidence (Section V).

Do typed decision models remove the need for labeled data? They remove the need for labels to specify a new decision. Testing that decision, and certifying an acceptance rule for a risk target, still need labeled evidence, which can sometimes be reused (Section IV).

Can Jev serve as a security gate? Only as one layer among several. Added context can move its decisions, and it can miss attacks that carry no explicit instruction, as can the open models tested with it (Section V). The design principles we recommend are to keep fetched content out of the gate, use allow, review and block outcomes, fix option names and order, and pin the model version (Section VI).

IX. CONCLUSION

Typed decision models bring classification, scoring and Boolean judgments into one interface whose questions are specified at call time. For new decisions, they address the goal that routing surveys state: a decision component that adapts without retraining. The routing approaches reviewed here still need candidate-specific evidence for new models, and risk targets still need labeled evidence. Independent studies report valid outputs and large savings against LLM judges, although a cheap generative cascade can be cheaper inside an agent loop. They also show accuracy gaps on hard tasks, calibration that varies by workload, and decisions that added context can move. We recommend three steps: 1) compare a typed model with the alternatives in Table III, including a label-probability readout on an open model and a cheap generative cascade; 2) measure both under the acceptance rule the deployment will use, on labeled data from its own traffic; and 3) record which labels, profiles and warmup batches can be reused when models, questions or constraints change.

Limitations. Most work specific to Jev is recent and not peer reviewed, and results can change between versions of a paper or of the model. Jev’s weights and training data are closed, so black-box measurements cannot separate model behavior from provider-side processing. The JevBench comparisons use one results snapshot without reported uncertainty estimates.

Our search covered public English-language sources and may have missed evaluations that have not appeared as papers or documented repositories.

REFERENCES

- [1] F. Standhartinger, “JevBench: A benchmark for Jev-class decision models, v1.4.2.2 results,” <https://github.com/fstandhartinger/jevbench/blob/main/results/v1.4.2.2/jevbench-v1.4.2.2-results.json>, 2026.
- [2] X. Yang, L. Li, H. Zhou, T. Zhu, X. Qu *et al.*, “Toward efficient agents: Memory, tool learning, and planning,” *arXiv preprint arXiv:2601.14192*, 2026.
- [3] C. Varangot-Reille, C. Bouvard, M. Ciancone, A. Gourru, M. Schaeffer, and F. Jacquenet, “Doing more with less: A survey on routing strategies for resource optimisation in large language model-based systems,” *Journal of Artificial Intelligence Research*, vol. 86, 2026. [Online]. Available: <https://doi.org/10.1613/jair.1.19801>
- [4] Y. Liu, “Evaluating system one models for agent security decisions: Reliability, calibration, and selective automation,” *arXiv preprint arXiv:2609.33401*, 2026.
- [5] Y. Sui, Y.-N. Chuang, G. Wang, J. Zhang, T. Zhang, J. Yuan, H. Liu, A. Wen, S. Zhong, N. Zou *et al.*, “Stop overthinking: A survey on efficient reasoning for large language models,” *arXiv preprint arXiv:2503.16419*, 2025.
- [6] S. Kadavath, T. Conerly, A. Askell, T. Henighan *et al.*, “Language models (mostly) know what they know,” *arXiv preprint arXiv:2207.05221*, 2022.
- [7] D. Kahneman, *Thinking, Fast and Slow*. New York, NY, USA: Farrar, Straus and Giroux, 2011.
- [8] TypeSafe AI, “TypeSafe AI documentation,” <https://docs.typesafe.ai>, 2026.
- [9] Nandakishor, “Laya: Non-autoregressive system 1 decision engine,” <https://github.com/NandhaKishorM/laya>, 2026.
- [10] J. Palmer, “Kev: Small Jev-like decision models you can train and run yourself,” <https://github.com/jaredpalmer/kev>, 2026.
- [11] Bespoke Labs and M. Sathiamoorthy, “Nimble,” <https://github.com/bespokelabsai/nimble>, 2026.
- [12] M. Marosi, “decider: One-pass typed decisions with calibrated probabilities,” <https://github.com/Mapika/decider>, 2026.
- [13] T. Lee, “SemIf (formerly OpenJev): Semantic ifs from open models,” <https://github.com/TheoLeeCJ/SemIf-OpenJev>, 2026.
- [14] J. Zhang, T. Yang, Y. Shi, and L. Wu, “AnyJev: Turn any LLM into a Jev-style decision model,” <https://github.com/nokia-applied-research/AnyJev>, 2026.
- [15] OpenAI, “DevDay 2026 Recap,” <https://openai.com/index/devday-2026-recap/>, Sep. 2026.
- [16] L. Tang and Y. Zheng, “Typed decision models: An early evidence audit and evaluation checklist,” *arXiv preprint arXiv:2609.32160*, 2026.
- [17] Awesome-Jev contributors, “Decisions, not tokens: A survey of typed, calibrated, and machine-native decision models from classical classifiers to Jev,” <https://github.com/youzizz1028/Awesome-Jev>, 2026.
- [18] M. Luo, “Jev and typed decision models: An empirical survey of calibration, selective control, and open implementations,” <https://github.com/Eurekaleo/awesome-jev-survey>, 2026.
- [19] Y. Moslem and J. D. Kelleher, “Dynamic model routing and cascading for efficient LLM inference: A survey,” *Transactions on Machine Learning Research*, 2026.
- [20] J. Gu, X. Jiang, Z. Shi, H. Tan, X. Zhai, C. Xu, W. Li, Y. Shen, S. Ma, H. Liu, S. Wang, K. Zhang, Z. Lin, B. Zhang, L. Ni, W. Gao, Y. Wang, and J. Guo, “A survey on LLM-as-a-judge,” *The Innovation*, vol. 7, no. 6, p. 101253, 2026. [Online]. Available: <https://doi.org/10.1016/j.xinn.2025.101253>
- [21] B. T. Willard and R. Louf, “Efficient guided generation for large language models,” *arXiv preprint arXiv:2307.09702*, 2023.
- [22] S. Kim, J. Suk, S. Longpre, B. Y. Lin, J. Shin, S. Welleck *et al.*, “Prometheus 2: An open source language model specialized in evaluating other language models,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.
- [23] Y. Liu, D. Iter, Y. Xu, S. Wang *et al.*, “G-Eval: NLG evaluation using GPT-4 with better human alignment,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.
- [24] Y. Geifman and R. El-Yaniv, “Selective classification for deep neural networks,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

- [25] B. Warner, A. Chaffin, B. Clavié, O. Weller, O. Hallström, S. Taghadouini, A. Gallagher, R. Biswas, F. Ladhak, T. Aarsen, G. T. Adams, J. Howard, and I. Poli, “Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference,” in *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2025, pp. 2526–2547. [Online]. Available: <https://aclanthology.org/2025.acl-long.127/>
- [26] L. Chen, M. Zaharia, and J. Zou, “FrugalGPT: How to use large language models while reducing cost and improving performance,” *Transactions on Machine Learning Research*, 2024. [Online]. Available: <https://openreview.net/forum?id=cSimKw5p6R>
- [27] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica, “RouteLLM: Learning to route LLMs from preference data,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [28] W. Jitritum, H. Narasimhan, A. S. Rawat, J. Juneja, C. Wang, Z. Wang, A. Go, C.-Y. Lee, P. Shenoy, R. Panigrahy, A. K. Menon, and S. Kumar, “Universal model routing for efficient LLM inference,” in *International Conference on Learning Representations (ICLR)*, 2026.
- [29] Y. Li, “Rethinking predictive modeling for LLM routing: When simple kNN beats complex learned routers,” *arXiv preprint arXiv:2505.12601*, 2025.
- [30] T. Feng, Y. Shen, and J. You, “GraphRouter: A graph-based router for LLM selections,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [31] H. Zhang, T. Feng, and J. You, “Router-R1: Teaching LLMs multi-round routing and aggregation via reinforcement learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.
- [32] C. Tran, S. Paracha, A. Hafeez, and S. Chen, “Arch-Router: Aligning LLM routing with human preferences,” *arXiv preprint arXiv:2506.16655*, 2025.
- [33] M. Yue, J. Zhao, M. Zhang, L. Du, and Z. Yao, “Large language model cascades with mixture of thoughts representations for cost-efficient reasoning,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [34] E. Chen, S. Wang, K. S. Chan, S. Wang, and C. G. Brinton, “Routing without training: Controllable-ratio LLM offloading via reliability gating,” *arXiv preprint arXiv:2607.20481*, 2026.
- [35] H. Li, Y. Zhang, Z. Guo, C. Wang, S. Tang, Q. Zhang, Y. Chen, B. Qi, P. Ye, L. Bai, Z. Wang, and S. Hu, “LLMRouterBench: A massive benchmark and unified framework for LLM routing,” in *Findings of the Association for Computational Linguistics: ACL 2026*. Association for Computational Linguistics, 2026, pp. 37 733–37 754. [Online]. Available: <https://aclanthology.org/2026.findings-acl.1881/>
- [36] TokenTrim, “jev-routing-experiment,” <https://github.com/TokenTrim/jev-routing-experiment>, 2026.
- [37] rishirajFS, “jev-router: Outcome-based evaluation of Jev as a cost-aware LLM router, on ground-truth benchmarks with three Claude tiers,” <https://github.com/rishirajFS/jev-router>, 2026.
- [38] V. Perifanis, N. Pavlidis, and S. Symeonidis, “SeLMRoute: Probabilistic semantic evidence for large language model routing,” *arXiv preprint arXiv:2609.34736*, 2026.
- [39] T. Z. Zhao, E. Wallace, S. Feng, D. Klein, and S. Singh, “Calibrate before use: Improving few-shot performance of language models,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.
- [40] H. Zhou, X. Wan, L. Proleev, D. Mincu, J. Chen, K. Heller, and S. Roy, “Batch calibration: Rethinking calibration for in-context learning and prompt engineering,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [41] C. Zheng, H. Zhou, F. Meng, J. Zhou, and M. Huang, “Large language models are not robust multiple choice selectors,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [42] W. Yin, J. Hay, and D. Roth, “Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.
- [43] I. Stepanov, M. Shtopko, D. Vodianytskyi, O. Lukashov, A. Yavorskyi, and M. Yaroshenko, “GLiClass: Generalist lightweight model for sequence classification tasks,” *arXiv preprint arXiv:2508.07662*, 2025.
- [44] H. Inan, K. Upasani, J. Chi, R. Rungta *et al.*, “Llama guard: LLM-based input-output safeguard for human-AI conversations,” *arXiv preprint arXiv:2312.06674*, 2023.
- [45] M. Hoover, V. Baherwani, N. Jain, K. Saifullah, J. J. Vincent, C. Jain, M. Rad, C. Bruss, A. Panda, and T. Goldstein, “DynaGuard: A dynamic guardian model with user-defined policies,” in *International Conference on Learning Representations*, 2026, pp. 87 822–87 859. [Online]. Available: https://proceedings.iclr.cc/paper_files/paper/2026/hash/8d2aad53fa148791ffd004a7c9cce5ad-Abstract-Conference.html
- [46] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, “On calibration of modern neural networks,” in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
- [47] A. N. Angelopoulos and S. Bates, “Conformal prediction: A gentle introduction,” *Foundations and Trends in Machine Learning*, vol. 16, no. 4, pp. 494–591, 2023. [Online]. Available: <https://doi.org/10.1561/22000000101>
- [48] A. N. Angelopoulos, S. Bates, A. Fisch, L. Lei, and T. Schuster, “Conformal risk control,” in *International Conference on Learning Representations (ICLR)*, 2024.
- [49] J. Jung, F. Brahman, and Y. Choi, “Trust or escalate: LLM judges with provable guarantees for human agreement,” in *International Conference on Learning Representations (ICLR)*, 2025.
- [50] Y. Ovadia, E. Fertig, J. Ren, Z. Nado, D. Sculley, S. Nowozin, J. V. Dillon, B. Lakshminarayanan, and J. Snoek, “Can you trust your model’s uncertainty? Evaluating predictive uncertainty under dataset shift,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.
- [51] sciethoon, “Does Jev know when it doesn’t know? Calibration of TypeSafe’s Jev on an unseen rule-based task,” <https://github.com/sciethoon/jev-ood-calibration>, 2026.
- [52] Y. Sun, J. Xu, J. Shi, and Z. Yang, “Type-safe is not error-free: A constrained decision head follows the option name, not the rubric bound to it,” *arXiv preprint arXiv:2609.26758*, 2026.
- [53] D. Rao and C. Callison-Burch, “JEV vs. LLMs as rubric judges: Cheaper, faster, and wrong in the same places,” *arXiv preprint arXiv:2609.29769*, 2026.
- [54] H. Ibrahim and Y. Zaki, “Evaluating decision models for text annotation in computational social science,” *arXiv preprint arXiv:2609.24574*, 2026.
- [55] R. Guo, Y. Liu, G. Deng, Y. Li, L. Zhao, Y. Wu, S. Chen, Y. Zhang, and L. Y. Zhang, “Just ask Jev: Reinforcement learning for calibrated decisions as a zero-shot detector of AI alignment failures,” *arXiv preprint arXiv:2609.29429*, 2026.
- [56] T. Wu and W. Y. B. Lim, “REFLEX with Jev for efficient selective control in LLM agents,” *arXiv preprint arXiv:2609.26532*, 2026.
- [57] T. Deuber, L. Sparrenberg, and R. Sifa, “Evaluating and benchmarking the system one model Jev,” *arXiv preprint arXiv:2609.37647*, 2026.
- [58] Y. Li, Y. Miao, R. Krishnan, and R. Padman, “JEV-as-a-judge: Accept when confident, escalate when unsure,” *arXiv preprint arXiv:2609.26550*, 2026.
- [59] J. Hu, H. Zhang, Y. Liu, Y. Zeng, L. Zeng, X. Wang, R. Wang, and L. Y. Zhang, “JevAdvBench: A benchmark and black-box attacks for reinforcement learning for calibrated decisions models,” *arXiv preprint arXiv:2609.31142*, 2026.
- [60] Z. Xu, “JevOut: Natural context can flip decision models,” *arXiv preprint arXiv:2609.30243*, 2026.
- [61] T. Wu and W. Y. B. Lim, “Decision hijacking: Prompt injection attacks on Jev’s typed probabilistic decisions,” *arXiv preprint arXiv:2609.28613*, 2026.
- [62] K. Li, Y. He, and Q. Li, “Beyond calibration: Do a typed-decision model’s probabilities obey the probability axioms?” *arXiv preprint arXiv:2609.33209*, 2026.
- [63] E. Debenedetti, I. Shumailov, T. Fan, J. Hayes, N. Carlini, D. Fabian, C. Kern, C. Shi, A. Terzis, and F. Tramèr, “Defeating prompt injections by design,” *arXiv preprint arXiv:2503.18813*, 2025.
- [64] L. Tunstall, N. Reimers, U. E. S. Jo, L. Bates, D. Korat, M. Wasserblat *et al.*, “Efficient few-shot learning without prompts,” *arXiv preprint arXiv:2209.11055*, 2022.
- [65] J. A. Hanley and A. Lippman-Hand, “If nothing goes wrong, is everything all right? Interpreting zero numerators,” *JAMA*, vol. 249, no. 13, pp. 1743–1745, 1983.