



Kai: Typed Decisions under Deterministic Policy

Hanzo AI Research

Hanzo AI

2026-10-22

Kai: Typed Decisions under Deterministic Policy

Results of a decision model and the policy that authorizes its actions

Hanzo AI Research*
Hanzo Industries Inc (Techstars '17)
San Francisco, California
<https://hanzo.ai>

2026-10-22

Abstract

Kai is a decision model. A caller states a question, the answers it admits and the evidence; Kai returns a calibrated probability over exactly those answers: one label of a choice, a yes or a no, or a level of a score. It writes no text. A deterministic policy, Nio, holds authority over action: the model recommends, the policy authorizes, and a model verdict can only make a policy verdict stricter. Kai serves `POST /v1/decisions` at `api.hanzo.ai` on CPUs, with answers that agree across CPU families to 2×10^{-5} . On a frozen harness of twelve decision tasks Kai averages 0.857 against 0.780 for Jev, a hosted decision model, and 0.675 for Laya, an open one, winning seven tasks and trailing on four. It keeps its accuracy as label sets grow from 4 to 150 options; its long-context release reads facts planted in states of up to 31,000 tokens at 0.73 to 0.80, where a model trained on short states reads at chance; and on Mind2Web its element accuracy is above the published Flan-T5-Base and GPT-4 baselines at a cost of \$0.00004 a step. Its typed grounding of action consequences, read once on a sealed suite of 14,832 scenes, is right on 99.94% of the scenes it grounds, and every scene whose harm has no exact count escalates to a person. We report where Kai trails as plainly: it beats neither Jev nor Clef on a public decision index, it trails on sealed transfer suites, it catches fewer jailbreaks than Jev at fewer false alarms, and without the policy it does not weigh the stakes of a trolley problem.

Contents

1	Introduction	3
2	The interface	3
3	Evaluation	4
4	Results	4
5	Limits	7
6	Conclusion	8
A	The wire	9

*Corresponding author: research@hanzo.ai

1 Introduction

Most decisions a system takes are typed. A ticket goes to one of ten queues, a message is or is not phishing, a request is urgent on a scale of three, an action is allowed, asked about or denied. A generative model can answer each of these by writing text, but the text must then be parsed, its probability is not the probability of the decision, and the same question asked twice can come back in two forms. Kai answers the decision itself. Given a state and a set of typed questions, it returns for each question a probability over exactly the answers the caller declared.

A model’s answer is not an authorization. Kai’s decisions that lead to action pass through Nio, a deterministic policy that holds authority over what is done: the model recommends, the policy authorizes, and a model verdict joined with a policy verdict on *allow* < *ask* < *deny* can only make it stricter. Where a decision turns on consequences, Kai reads the consequences into typed state and Nio decides from that state alone; where the evidence does not establish them, the decision goes to a person.

This paper reports what Kai achieves against public benchmarks and the models they are usually read with. Section 2 states the interface. Section 3 gives the evaluation protocol and the baselines. Section 4 gives the results, where Kai leads and where it trails, and Section 5 what they do not show.

2 The interface

A request holds a state (text or JSON), between 1 and 100 questions and, optionally, a policy. A question is one of three types:

- a *choice* over two or more labels, answered by a probability for each label;
- a *yes/no* question, answered by the probability of yes;
- a *score* over one or more ordered levels, answered by a probability for each level and its expected level.

Every answer is a distribution over the declared answers and nothing else, so it is read without parsing and compared without a judge. Unknown keys, repeated keys and malformed questions are refused with the path of the fault. Appendix A shows a request of each kind.

Policy. A request may carry a policy instead of leaving the decision to the model. Under the harm policy each course of an action names the people it would harm, either stated by the caller or read by Kai as typed answers to registered questions (what lies on the path, and how many people). Nio takes the course that harms fewest, keeps the default course on a tie, and escalates to a person when any count is unknown, inexact or read below the policy’s confidence floor of 0.9. A yes/no answer the policy governs is set by the rule, never by the model, and whatever acts on a verdict passes one gate that refuses any action the verdict does not take. A verdict is advice to a person or to a system that checks it again; Kai’s decisions are not used to actuate physical systems.

Serving. Kai runs on CPUs in full precision. Two CPU families, an arm64 host and an x86-64 host, give the same answers to 2×10^{-5} on every probability and the same policy outcome on every request measured (Section 4.8), so an answer read in evaluation is the answer production serves. A request is answered the same way whatever else is queued.

3 Evaluation

Protocol. Every number is a kept run read through the same path production serves, `kai serve` on a CPU, unless a table says otherwise. A benchmark’s test items never enter training: every item of every benchmark in this paper, its normalized text and its near duplicates are held out of every training set by an exact and near-duplicate barrier. A sealed suite is written and hashed before the model it measures is frozen, read once, and spent after its read. Comparisons are paired on the same items: accuracy by McNemar’s exact test, other statistics by a paired bootstrap, with Holm’s correction over a table’s family of tests. Intervals are 95%.

Baselines. *Jev* 1.13 is a hosted decision model served through OpenRouter’s Decisions API [7]; its numbers are its own answers to the same requests, recorded once. *Laya* is an open decision model [2], run from its released weights. *Clef*’s and *Von*’s numbers are as their authors recorded them on the same suites. Where a public paper reports a baseline, we cite it and say so.

Releases. Kai is one model line served under one endpoint. A table reports the release that was read on it: the production model, its long-context release, or the production base with a task head added for one benchmark. A task head reads the base’s own pass of a request and changes no other answer.

4 Results

4.1 Decision tasks

The frozen decision harness asks 11,099 questions over 9,499 states drawn from twelve tasks: news topics [10], emotions [8], banking intents [1], support triage, spam, phishing, jailbreaks [6], toxicity, retrieval relevance, routing, typed decisions, and intents in 51 languages [4]. Table 1 gives accuracy. Kai averages 0.857 against Jev’s 0.780 and Laya’s 0.675. Against the better of the two baselines, within 0.005 a tie, Kai wins seven tasks, ties one and trails on four.

4.2 Many labels, and labels never seen

A decision often has more than a handful of answers. Table 2 reads accuracy as the label set grows, from four labels to Banking77’s 77 and CLINC150’s 150 [1, 5]. Kai holds 0.908 at 77 and 150 labels, ahead of Jev up to 77 and behind it at 150. On a public zero-shot pilot, 100 items a dataset with Jev’s own request shape, Kai reads AG News at 0.940 and emotions at 0.750 against Jev’s published 0.910 and 0.480, and Banking77 with 72 sentence-long labels at 0.810 against 0.870; 35 of the pilot’s 300 items also appear in the harness.

4.3 Long states

A planted-fact task hides one fact at a swept position in real text of a given length, with decoys in 60% of states, and asks a choice, a score or a yes/no question about it: 120 states a cell, chance about 0.33. Kai’s long-context release reads it through the served CPU path (Table 3). A model of the same size trained on states of up to 1,024 tokens reads at chance from the first cell.

task	Kai	Laya	Jev
AG News	0.943	0.950	0.860
DAIR emotion	0.935	0.595	0.603
Banking77	0.907	0.425	0.835
support triage	0.443	0.502[†]	0.365
email spam	0.998	0.998	0.978
phishing	0.990	0.983 [†]	0.900
jailbreak	0.920	0.705	0.940
toxicity	0.802	0.530	0.662
RAG relevance	0.675	0.625	0.620
routing	1.000	0.639	0.977
typed decisions	0.759	0.766	0.736
MASSIVE, 51 languages, macro	0.909	0.382	0.890
mean of the twelve	0.857	0.675	0.780

Table 1: The frozen decision harness, accuracy. Kai is the production base; its yes/no rows were read on the base’s own yes/no answer, and on 951 held-out items of those tasks the production model reads 0.883 against Jev’s 0.875. [†]Laya’s support-triage and phishing items are drawn from data its builder marks as trained on.

labels	4	16	77	150
Kai	0.968	0.935	0.908	0.908
Jev	0.905	0.853	0.843	0.940

Table 2: Accuracy as the label set grows (400 questions a cell; 77 is Banking77, 150 is CLINC150).

4.4 Web agents

Mind2Web [3] asks, at each step of a real website task, which of the page’s candidate elements to act on and how. Kai answers each step as a choice over the 50 candidates the benchmark’s ranker supplies, with a task head trained on Mind2Web’s training split. Table 4 reads its element accuracy and step success on the three test splits. Kai’s element accuracy is above the published Flan-T5-Base and GPT-4 numbers on every split and below Flan-T5-XL’s. Kai writes no text, so a step that types a value cannot count as a full success, which holds its step success below Flan-T5-Base’s. On the same 300-step samples, Claude Haiku 5.5 and Claude Opus 5.5 given the same candidates read 47.3 / 43.3 / 50.0 and 59.7 / 58.0 / 60.7 element accuracy against Kai’s 51.3 / 41.3 / 40.7, at \$0.0003 and \$0.011 a step against Kai’s \$0.00004.

4.5 Tools

An agent must pick the right tool for a request, and know when no tool fits. On held-out lists of candidate tools drawn from public tool-use datasets and Hanzo’s own catalog, 5,658 ranked cases, Kai’s agentic release ranks the right tool with nDCG@10 0.894 [0.889, 0.898] and puts it first on 0.758 [0.747, 0.769] of cases, against BM25’s 0.781 and 0.541. Asked whether to call a tool at all, it calls on 0.122 [0.087, 0.168] of the cases where none applies while keeping 0.919 of the cases where one does.

state length (tokens)	900	4,000	8,000	16,000	31,000
Kai, long-context release	0.800	0.758	0.733	0.792	0.733
the same base trained to 1,024 tokens	0.333	0.308	0.375	0.350	0.300

Table 3: Planted-fact accuracy by state length. The long-context release is read on an x86-64 CPU; the short base was read on a GPU.

	element accuracy			step success		
	task	website	domain	task	website	domain
Kai, with a task head	48.8	41.5	39.1	37.5	28.6	29.2
Flan-T5-Base (published)	43.6	32.1	33.9	41.0	29.5	31.6
Flan-T5-XL (published)	55.1	42.0	42.1	52.0	38.9	39.6
GPT-4, 50 tasks (published)	41.6	35.8	37.1	36.2	30.1	26.4

Table 4: Mind2Web cross-task, cross-website and cross-domain splits, percent. Kai’s cross-domain split is its first 2,000 steps.

4.6 Safety

Table 5 reads the production model on two sets. On InjecAgent [9], instructions planted in a tool’s output for an agent its user never gave, Kai catches 55% of the attacks at 2.4% false alarms. On a pool of 1,163 real jailbreak and injection attacks and 1,006 human-written benign prompts, Jev ranks attacks better and catches far more of them at 0.5; Kai raises under a third of Jev’s false alarms.

4.7 Consequences under policy

Under the harm policy, Kai reads for each course of an action what lies on its path and how many people, and Nio decides. This grounding was qualified once, on a sealed suite of 14,832 generated scenes in two settings no candidate had trained on, at the policy’s floor of 0.9 (Table 6). 5,680 of the scenes state a harm with no exact count (a crowd past the numerals, a hedged count, infrastructure others depend on, a vague harm) and must escalate; the rest must ground. Of the scenes it grounded Kai was right on 99.94%, with 5 wrong courses in 14,832 scenes, and it grounded 2 of the 5,680 it should not have. On 9,600 further scenes built only of unusable evidence (two possible counts, ranges, corrupted figures, crowds, hedges and dependent infrastructure) it grounded none. One person read the same whether described as a stranger, an employee, a chief executive or the system’s own developer, on 480 of 480 variants, and a server that keeps someone alive escalated on 400 of 400 scenes where an isolated server read as property.

On a public trolley-problem benchmark of 5,240 setups, Kai’s own yes/no answer does not weigh the stakes: it pulls the lever in 650 of the 672 setups where pulling kills more than it saves, against Jev’s 69 and GPT-6 Luna’s 143. Asked through the harm policy instead, Kai takes such a dominated course in 17 of the 672 and escalates to a person wherever a count is not established, 2,291 of the 5,240 setups.

4.8 Serving

The production model serves on CPUs in full precision. Its release was read on an arm64 host and an x86-64 host before it rolled (Table 7), then canaried through the public API: every answer

	Kai	Jev
InjecAgent: attacks caught, of 250	138	–
InjecAgent: benign flagged, of 207	5	–
real attacks: AUROC	0.725	0.896
real attacks: caught at 0.5	26.9%	77.4%
human-written benign: flagged at 0.5	3.6%	13.0%

Table 5: Safety reads of the production model.

scenes	14,832
grounded (both courses read at the floor)	60.8%
right, of the grounded	99.94%
wrong courses, of all scenes	5 (0.03%)
grounded where the harm has no exact count	2 of 5,680
unusable-evidence scenes grounded	0 of 9,600
one person read alike across four roles	480 of 480
life-support infrastructure escalated	400 of 400

Table 6: The sealed qualification of the production model’s typed grounding, at the floor of 0.9.

matched the x86-64 reference to 1.5×10^{-6} .

4.9 Where Kai trails

On a public decision index of 38 benchmarks, Kai beats neither Jev nor Clef on any of the 32 rows it was read on. By area, Kai, Jev and Clef read arts 0.35, 0.61 and 0.63; knowledge 0.24, 0.66 and 0.66; language 0.38, 0.81 and 0.83; retrieval 0.51, 0.72 and 0.79; tools 0.14, 0.77 and 0.83. The knowledge and language rows ask what a compact decision model does not hold, and Hanzo answers them with its generative models instead. On sealed transfer suites of decision tasks Kai never trained toward, its pooled accuracy is 0.440 against Laya’s 0.576 and Von’s 0.646 and 0.711; the hardest tier, long policies and multi-hop reasoning, is at chance for every model. On jailbreaks Jev catches far more at 0.5 (Table 5), and on the trolley benchmark Kai’s own answer does not read the stakes (Section 4.7).

5 Limits

Advice, not actuation. A policy verdict is advice to a person or to a system that checks it again. Kai’s grounding was qualified on generated scenes in settings it had not seen, not on incidents in the field, and its own yes/no answer does not weigh stakes; Kai is not used to actuate physical systems, and a consequential action stays with a person when the evidence does not establish its counts.

Counts. The production policy reads counts from 0 to 100 and escalates anything past them or stated inexactly. Reading an exact count of any size, bound to the right course, from the model alone is the next release’s work; no suite reported here measures it.

Breadth. Kai decides; it does not reason through knowledge-heavy or multi-hop tasks, where it trails generative models (Section 4.9). Its transfer to decision tasks unlike its training is below the best open model’s, and its jailbreak recall is below Jev’s.

yes/no answers across CPU families: largest difference	1.8×10^{-5}
yes/no answers across CPU families: decisions that differ	0 of 955
typed policy requests across CPU families: outcomes that differ	0 of 400
production canary: answers within 1.5×10^{-6}	16 of 16
production request of two questions: server time, median	209 ms
yes/no request, x86-64, four threads: median / 95th / 99th	172 / 1,122 / 1,570 ms
policy request of seven questions, x86-64: median	947 ms
peak resident memory of a server	3.31 GB

Table 7: Serving the production model.

Samples. Some reads are small: 100 items a dataset in the zero-shot pilot, 300 steps a split for the hosted comparison on Mind2Web, 254 cases where no tool applies. Their intervals, not their points, are the claim.

6 Conclusion

Kai answers decisions as distributions over declared answers, and Nio, a deterministic policy, alone authorizes action. On a frozen harness of twelve decision tasks Kai leads a hosted and an open decision model on average; it scales to large label sets, reads long states in its long-context release, picks web elements and tools at a fraction of a generative model’s cost, and grounds the consequences of an action with an error rate of 0.03% of scenes while escalating every harm it cannot count. It trails on broad knowledge, on transfer to unlike tasks and on jailbreak recall, and these numbers say so. Kai serves at `api.hanzo.ai`, with the same answers on every CPU family it runs on.

References

- [1] Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. Efficient intent detection with dual sentence encoders. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, 2020.
- [2] Convai Innovations. Laya: a multilingual, non-autoregressive decision engine. <https://huggingface.co/convaiinnovations/laya>, 2026. Version 0.3.20, Apache-2.0.
- [3] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2Web: Towards a generalist agent for the web. In *Advances in Neural Information Processing Systems*, 2023.
- [4] Jack FitzGerald, Christopher Hench, Charith Peris, et al. MASSIVE: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics*, 2023.
- [5] Stefan Larson, Anish Mahendran, Joseph J. Peper, Christopher Clarke, Andrew Lee, Parker Hill, Jonathan K. Kummerfeld, Kevin Leach, Michael A. Laurenzano, Lingjia Tang, and Jason Mars. An evaluation dataset for intent classification and out-of-scope prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 2019.
- [6] Zi Lin, Zihan Wang, Yongqi Tong, Yangkun Wang, Yuxin Guo, Yujia Wang, and Jingbo Shang. ToxicChat: Unveiling hidden challenges of toxicity detection in real-world user-AI conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023.

- [7] OpenRouter. Decisions API (alpha): POST /api/alpha/decisions. <https://openrouter.ai>, 2026. Accessed 2026-09-24.
- [8] Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. CARER: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.
- [9] Qiusi Zhan, Zhixiang Liang, Zifan Ying, and Daniel Kang. InjecAgent: Benchmarking indirect prompt injections in tool-integrated large language model agents. In *Findings of the Association for Computational Linguistics: ACL 2024*, 2024.
- [10] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*, 2015.

A The wire

A request with one question of each type. The state is invented for this example.

```
{
  "model": "kai",
  "state": "Hi, we were billed twice for March. Please refund the duplicate today or we will
    cancel our plan.",
  "questions": {
    "department": {
      "type": "choice",
      "instructions": "Which department should handle this?",
      "criteria": {
        "billing": "invoices, payments, refunds",
        "technical": "bugs, outages, system errors",
        "other": "everything else"
      }
    },
    "urgency": {
      "type": "score",
      "instructions": "How urgent is this?",
      "criteria": ["not urgent", "soon", "blocking"]
    },
    "churn_risk": {
      "type": "boolean",
      "instructions": "Does the user threaten to cancel or leave?"
    }
  }
}
```

The answer to `department` carries choice, probabilities over the three labels, `confidence` and `answer_confidence` (its top probability); `urgency` carries the expected level `score`, a legend and the same summaries; `churn_risk` carries its probability alone. The `routing` object names the model that answered and its weights' SHA-256.

A typed harm request. The scene is invented for this example and belongs to no suite. The four typed questions carry the registered wordings; the policy names them for the two courses and governs the yes/no question `act`.

```
{
  "model": "kai",
```

```

"state": "You control the brake of a gantry rolling along a greenhouse aisle. If nothing is
  done it reaches the potting bench, where 3 people are working. If you throw the brake it
  stops in the empty seed store.",
"questions": {
  "default.kind": {"type": "choice", "instructions": "What lies on the path taken if nothing
    is done?",
                  "criteria": ["people", "nobody", "asset", "unquantified", "infrastructure
    "]},
  "default.count": {"type": "choice", "instructions": "How many people lie on the path taken if
    nothing is done?",
                   "criteria": ["0", "1", "2", "...", "100", "more"]},
  "action.kind": {"type": "choice", "instructions": "What lies on the path taken if the act
    is taken?",
                 "criteria": ["people", "nobody", "asset", "unquantified", "infrastructure
    "]},
  "action.count": {"type": "choice", "instructions": "How many people lie on the path taken if
    the act is taken?",
                  "criteria": ["0", "1", "2", "...", "100", "more"]},
  "act": {"type": "boolean", "instructions": "Should the brake be thrown?"}
},
"policy": {
  "profile": "nio/harm01",
  "default": "leave",
  "courses": {
    "leave": {"people": {"kind": "default.kind", "count": "default.count"}},
    "throw": {"people": {"kind": "action.kind", "count": "action.count"}}
  },
  "answers": {"act": "throw"}
}
}

```

The count labels are written out in full in a real request ("0" to "100", then "more"); the listing abbreviates them. The response's policy object holds the profile, the course taken or none, its basis (fewest, default, unknown or tied), whether it escalates, the reading of each course (kind, its probability, count, its probability, people, and why), and the questions it governs. The answer to act is set by the rule, not by the model: 1 or 0 with probability of acting 1, or 0.5 with probability of acting 0 under escalation.