LogitTrail: An Inspectable Local Interface for Typed Decisions from Frozen Language Models

Yuki Oshio
UPHASH Inc.
oshio@uphash.net

Abstract

Many language-model applications need a small typed decision rather than free-form text. LogitTrail is an open-source local interface that maps a state and explicit criteria to a categorical choice, a true-candidate probability, or an expected ordinal stage. It uses candidate next-token logits from a frozen language model, with no trained decision head. A browser demonstration exposes typed values, candidate distributions, probability semantics, and request/response JSON. A 4,050-request matched comparison finds identical direct-readout and native one-token outputs in all 1,350 pairs, without an established latency advantage. Two subsequent studies across three checkpoints use 7,600 and 14,400 requests to examine presentation sensitivity and probability averaging. Reduced sensitivity can coexist with incorrect or nearly constant predictions, while averaging costs multiple model calls. The contribution is a runnable demonstration and auditable system evaluation, with explicit distinctions among type validity, candidate concentration, task quality, and computation cost.

1 Introduction

A workflow may need a route, an urgency score, or a stage on an explicit rubric. Returning a paragraph and then recovering that value creates an interface problem: the application must validate a result whose useful content is small. A typed interface makes the contract explicit, but a valid type alone says nothing about whether the decision is right. Similarly, a model can assign almost all candidate probability to an incorrect answer.

LogitTrail (formerly Mini Jev) demonstrates this distinction through a local browser interface and an HTTP/Python API. Users edit a shared state and three questions, run the model, and inspect the resulting values alongside candidate distributions. The interface is inspired by TypeSafe's Jev (Almeida, 2026; TypeSafe, 2026b); this independent implementation makes no claim to reproduce that system's architecture, training, calibration, compatibility, or performance.

Developers can inspect local typed decisions before integration. The mechanism is conventional: frozen candidate scoring, restricted softmax, and typed response construction. Verbalizers, logit-based selection, and scoring APIs are established (Schick and Schütze, 2021; Ma et al., 2025; LMQL contributors, 2026). Our contribution combines inspectable local implementation and auditable evaluation of quality, presentation sensitivity, and cost. Neither scoring nor typed probability interfaces are new.

Schema validity, regression checks, task quality, and paired timing answer different questions. Our matched study does not establish a latency advantage over native one-token selection. Source, installation, data documentation, and evaluation artifacts: github.com/UpHash-Network/mini-jev. Video: approximately 145-second captioned live recording.

2 System and Typed Contract

2.1 From a state to typed values

A request contains a shared `state` and named questions. Each question supplies a type, natural-language instructions, and candidate criteria. The service returns a semantic label, a candidate distribution, and a type-specific value:

- **Choice** returns the key of the most likely candidate. Keys are canonically sorted before prompting.
- **Noul** exposes the probability of the `true` candidate in a two-candidate false/true decision.
- **Score** returns the expected zero-based stage of an ordered rubric; the most likely stage is also available as the label.

The distinction between Score's expectation and its hard label matters. An expectation of 1.6 on stages 0–3 is a probability-weighted position, not an additional model-generated stage or a calibrated utility. Its interpretation assumes equally spaced stage indices; users must decide whether that fits their rubric.

Let z_i denote the next-token logit for candidate $i \in \{0, \ldots, K-1\}$, with K candidates and temperature $T > 0$. The service computes

$$p_i = \frac{\exp(z_i/T)}{\sum_{j=0}^{K-1} \exp(z_j/T)}, \qquad s = \sum_{i=0}^{K-1} i p_i. \tag{1}$$

These are probabilities conditional on the selected candidate set. They are neither probabilities over all valid textual answers nor guarantees that the criteria are exhaustive. Changing the criteria changes the question as well as the normalization. The optional scalar temperature preserves the maximum-logit label, but can change Noul and Score values.

The interface also displays entropy concentration,

$$c = 1 - H(p)/\log K, \qquad H(p) = -\sum_i p_i \log p_i. \tag{2}$$

A concentrated distribution has c near one. The display identifies this quantity as concentration rather than estimated accuracy. The response includes probability-semantics and calibration metadata, so downstream applications can retain the same distinction.

2.2 Local execution

The released inference configuration uses Qwen3.6-35B-A3B in Q4_K_M GGUF form (Qwen Team, 2026; ggml-org, 2026), through a source-built `llama.cpp` helper (ggml-org and contributors, 2026). The model and its output projection remain frozen; there is no trained decision head. The helper performs the full vocabulary projection and then gathers candidate logits. It does not implement a selected-row projection optimization.

The prompt repeats the task twice, following the use of repetition for non-reasoning inference (Leviathan et al., 2025). Choice and Noul use letter labels; Score with at most ten stages uses digits after a prefilled answer-object prefix. Larger

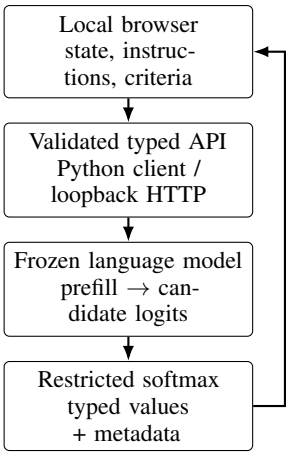

Figure 1: The live demonstration uses the released typed API. It displays model-derived candidate distributions and typed values; no generated explanation or simulated prediction is inserted.

Score rubrics use letters. The implementation checks that every candidate is a distinct single token at the actual prefix boundary. The API accepts 2–26 candidates; quality evaluation covers 2–8, while larger candidate sets have functional coverage only.

A multi-question request executes its questions sequentially. Model state is cleared between questions: they share the supplied state text, but not a cached prefill. Complete inputs exceeding 2,048 tokens are rejected rather than silently truncated. Schema validation and deterministic response construction prevent free-form output from crossing the API boundary; these software properties do not prevent semantic errors.

2.3 Relationship to one-token decoding

Restricted softmax readout has the same mathematical candidate distribution as a sampler that masks all non-candidates, provided the model prefix, logits, candidate IDs, temperature, and other processors match. Greedy labels then agree unless tie-breaking differs; stochastic sampling agrees in distribution, not necessarily in realized samples. This is an equivalence under specified conditions, not a new inference theorem or learning method.

In particular, selecting a first token after prefill need not require another model forward pass. Avoiding a generated token therefore does not itself establish a speedup. The matched study in Section 4 tests this point using an actual native one-token sampler, rather than comparing direct readout only with a long textual answer.

3 Demonstration and Access

The browser runs locally and connects through a small Python bridge to the existing model API. It requires no cloud inference service. A user starts with a shared state and three editable questions: routing through Choice, urgency through Noul, and priority through Score. The question instructions and criteria are visible, making the decision contract available for inspection before inference.

The user defines the state and criteria and selects *Run decisions*. Candidate probabilities distinguish a hard label from its distribution; Score's expected and most likely stages can differ. Rerunning after edits explores behavior, but changes the input and is not a controlled accuracy experiment.

The page also displays entropy concentration, elapsed time, and copyable request/response JSON. The bars show relative candidate mass, the JSON exposes the contract, and elapsed time describes the live request; none is a reference answer. Editing marks prior results stale. There is no replay mode: an unavailable model API produces an error, not a canned prediction.

Installation begins with the repository's native build script and pinned model downloader. Users build the helper with `build_native.sh`, download the model with `fetch_native_model.py`, and start the API with `./run.sh -without-calibration`. Starting `python3 -m demo.server -port 8766` then opens access at `http://127.0.0.1:8766/`. The repository supplies complete commands and a Python client example. Fresh builds start at $T = 1$; historical calibration artifacts are bound to their recorded runtime and should not silently transfer to a different build.

The tested environment is an Apple M5 Pro with 64 GB unified memory and macOS 26.4, using Metal and Accelerate. The model file is approximately 20.4 GB, in addition to runtime memory. This is a local workstation demonstration, not a claim of lightweight deployment or a demonstrated minimum-memory requirement. Source is distributed; model weights and compiled native binaries are not bundled.

4 Evidence Behind the Demonstration

4.1 Evaluation panels and provenance

Table 1 separates the evidence sources. Repeated requests are not additional quality examples, and the external tasks do not share one accuracy metric.

Evidence	Size	Interpretation
Local regression	2,400 items	Self-authored Japanese; 800 per type
Expanded external	600 items	JCoLA, JSTS, JCQA; 200 each
Matched runtime	4,050 requests	150 local $\times$ 5 repetitions, plus 600 external; 3 modes
Presentation study	7,600 requests	Same 600 external items; 3 checkpoints
Cyclic averaging	14,400 requests	600 additional external items; 3 checkpoints

Table 1: Evaluation accounting. The last two studies use 1,200 unique source items in total. Measured counts exclude synthetic checks and warm-ups. JCQA abbreviates JCommonsenseQA.

The historical local suite contains 2,220 generated items from 45 template families and 180 individually AI-authored items; its legacy `manual` field does not mean human annotation. Authoring used earlier errors, so this is regression material rather than a blind test. All 2,400 outputs were valid, but 162 top labels disagreed with authored reference labels: structural validity does not establish correctness. Temperature fitting did not improve all calibration and score-error metrics.

The expanded sample has 200 public-development examples each from JCoLA (Someya et al., 2024), JSTS, and JCommonsenseQA (Kurihara et al., 2022). Hash selection and rubrics were frozen before inference, without independent preregistration. Six JSTS rows share text or image provenance with an earlier 300-item JNLI pilot. Public development data may have appeared in model training; these are not independent hidden-test evaluations.

4.2 Matched runtime and parity

All three conditions used one resident model and a common research HTTP response containing a type and semantic label. Direct readout and native one-token greedy selection used identical complete prompts and candidates. The JSON condition actually generated a grammar-constrained object, with a serialization-specific prompt. Its timing comparison therefore includes prompt and prefill differences as well as multi-token generation; it does not isolate sampler overhead.

The schedule contained $(150 \times 5 + 600) \times 3 = 4{,}050$ measured requests, plus 21 synthetic warm-ups. All completed with valid responses and no

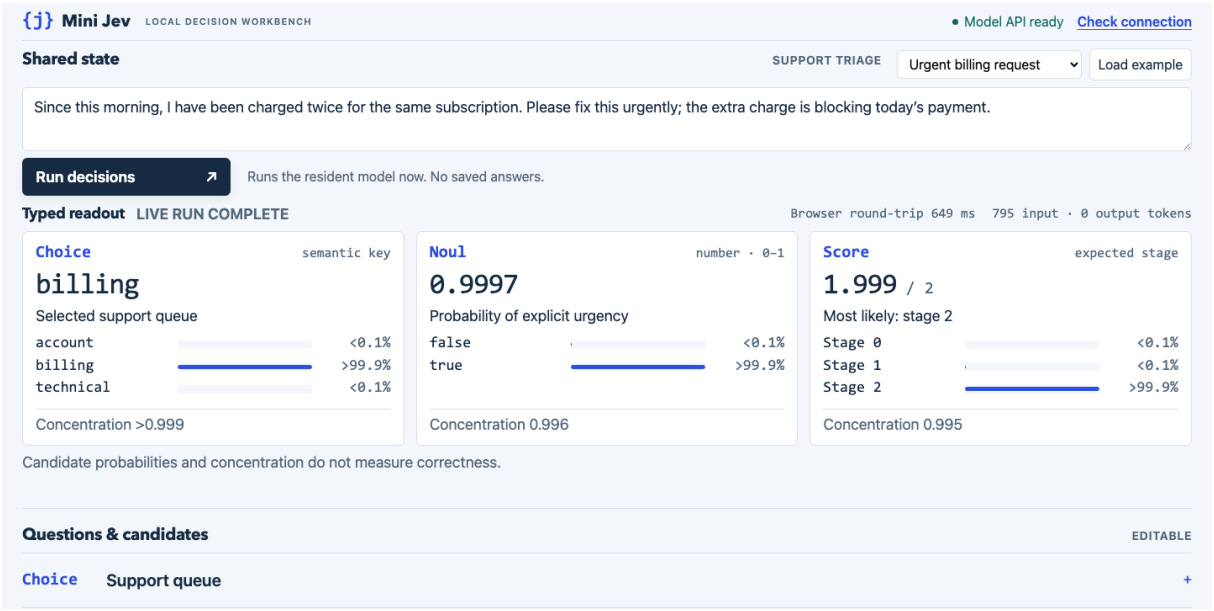

Figure 2: Recorded browser demonstration under the former Mini Jev name: an editable state, all three typed outputs, and their candidate distributions. Editors continue below the results. This English example illustrates the interface, not English task quality.

Comparator − direct	Difference	95% interval
One token	+0.0898	[−0.7483, 0.8449]
JSON	+160.2958	[138.1189, 177.0587]

Table 2: Primary paired HTTP latency differences in milliseconds. Intervals resample observed local family/tag clusters; JSON uses a different serialization prompt.

recorded HTTP failures. Mode order was randomized within each item/repetition. Timing covered serialization through loopback HTTP parsing, including native audit transfer but excluding trace-file writes. These are research-harness timings, not browser rendering times.

Direct and one-token conditions matched prompts, token sequences, candidate IDs, logits, and labels in all 1,350 paired observations. Maximum observed candidate-logit difference was zero; there were no maximum-logit ties. The primary latency statistic is the median across 150 local items of the within-item difference between five-repetition mean latencies. It is not a difference between marginal medians.

Table 2 gives 10,000-replicate whole-family/tag percentile intervals. They describe sensitivity to the observed cluster composition, conditional on this model, machine, selected items, and session. They do not estimate between-machine or independent-session variability. The one-token interval crosses zero: this run provides no evidence of a material direct-readout latency advantage. JSON was slower under its different serialization requirements, with task-dependent quality changes.

4.3 External task-specific results

Direct readout and one-token selection have identical external labels. JCoLA reached 86.0% accuracy and MCC 0.5708; JSON reached 82.0% and MCC 0.5160. The selected sample contains 158 acceptable sentences, so an always-acceptable rule already achieves 79.0% accuracy and has undefined MCC. Reporting only accuracy would obscure this imbalance.

JCommonsenseQA accuracy was 95.0% for direct/one-token and 95.5% for JSON, a difference of one example. JSTS retains its continuous reference scores in $[0, 5]$: direct expected-stage MAE at $T = 1$ was 0.5065. JSON has no comparable probability vector, so we do not assign it an expected-stage metric. The separate hard-stage comparison gives MAE 0.5480 for direct/one-token and 0.5650 for JSON. We do not round the gold scores into invented classes or pool these tasks into one accuracy.

4.4 Presentation sensitivity and averaging

Two subsequent studies add Qwen2.5-0.5B-Instruct and Qwen2.5-1.5B-Instruct to the native

35B checkpoint. The first uses 7,600 requests on the same 600 external items. All 1,800 exact-input repeat pairs have identical logits and probabilities, yet display reversal changes native JCQA labels on 11/200 items. Conversely, the 1.5B checkpoint has zero JCoLA flips while always predicting acceptable: stability alone is insufficient.

The second freezes 600 additional public-development items before its 14,400 requests. It excludes prior IDs and literal text reuse, and selects JSTS pairs from previously unused sentence/image components. However, 194/200 JCoLA items share bibliographic sources with earlier evaluations. Both studies complete without inference errors; each excludes 57 synthetic checks/warm-ups.

We average semantic candidate probabilities at $T = 1$ across cyclic rotations of complete option lines, keeping output-token–meaning bindings fixed. Fixed-binding transformations and cyclic aggregation have precedent (Wei et al., 2024; Zheng et al., 2024); this is not a new algorithm or a reproduction of PriDe. Table 3 compares canonical-single and forward-cyclic accuracy, alongside single/reversed and forward/reverse-cyclic label disagreement. Lower disagreement does not consistently improve accuracy. The 0.5B cyclic JCoLA output is always acceptable (78.5% accuracy, undefined MCC). On JSTS, 0.5B and 1.5B cyclic hard labels occupy one stage on 199/200 and 200/200 items, respectively. These marginals expose collapse hidden by type validity or stability.

One cyclic answer costs 2, 5, or 6 model calls for JCoLA, JCQA, or JSTS; the single baseline reuses a pool member. Binary forward/reverse pools contain the same two calls, so their equality is structural. There is no equal-call repeat control in this averaging study. The small checkpoints use Transformers/MPS float32; the native model uses quantized Metal inference. Architecture, precision, and backend differ, precluding a model-size or cross-backend speed conclusion. These are descriptive system checks; fuller distributions, grouped intervals, and call accounting accompany the artifacts.

4.5 Reproduction and framework integration

The versioned source/evidence package at github.com/UpHash-Network/mini-jev records the three studies' 26,050 requests, protocols, selections, hashes, analyzers, and AI-agent audits.

Checkpoint	Accuracy (%)	Flip rate (%)
Qwen2.5 0.5B	$39.5 \rightarrow 36.0$	$65.5 \rightarrow 30.5$
Qwen2.5 1.5B	$80.0 \rightarrow 83.5$	$26.0 \rightarrow 12.5$
Qwen3.6 35B-A3B	$98.5 \rightarrow 98.0$	$4.5 \rightarrow 1.5$

Table 3: JCQA on the additional 200-item panel: single $\rightarrow$ cyclic point estimates. Flip rate measures orientation disagreement, not error rate. Native 35B uses Q4_K_M. These are not equal-compute comparisons.

Clean-directory CPU reanalysis reproduced all 56 derived files byte-for-byte. A fresh native build served all three types using a cached, hash-verified model on the same Mac. This validates reconstruction with source/model cache reuse, not a second-machine or fresh-download replication.

Separate synthetic diagnostics test integration boundaries. With the same checkpoint, complete prefixes and candidate IDs, unmodified LMQL 0.7.3 scoring through a custom native backend matched all 12 direct-readout labels. Maximum probability difference was 2.63×10^{-5}: three cases exceeded the prespecified 10^{-5} tolerance. Different native scoring paths therefore did not establish exact numerical parity. ChainForge 0.3.7.4's registration/dispatch routes, with a custom provider calling the released API, preserved all eight typed-answer pairs across four request pairs (maximum probability difference zero); three invalid inputs were rejected. This checks backend interoperability, not the browser flow editor, human usability, or task quality. These diagnostics are separate from the three evaluation panels. Inference replication still requires upstream data, pinned models, and compatible runtimes; hardware/build equivalence is not guaranteed.

5 Related Work and Scope

Candidate and ordinal scoring. Verbalizers (Schick and Schütze, 2021), decoding-free selection (Ma et al., 2025), Qwen3 reranking (Zhang et al., 2025; Qwen Team, 2025), and G-Eval's weighted scores (Liu et al., 2023) precede our readouts. Zhuang et al. (2024) compute expected relevance from normalized fine-grained label log-likelihoods in zero-shot ranking. Wang et al. (2026) use grade-token expectations for job-candidate ranking and fine-tune with a combined mean-squared-error and cross-entropy loss. Score shares this established expectation; frozen-weight scoring is also not new. Our contribution concerns typed inspection and system evaluation, not rank-

ing gains or a new scoring rule.

Structured interfaces. LMQL exposes continuation scoring and selection (LMQL contributors, 2026); multi-token scoring need not equal our single-token readout. Outlines, Guidance, and grammar-constrained decoding support structured generation (dottxt and Outlines contributors, 2026; Guidance contributors, 2026; Geng et al., 2023). TypeSafe's adapter maps general LLM outputs to a Jev-style contract (TypeSafe, 2026a); generated probability values differ from token-derived candidate probabilities. Our JSON experiment is a serialization comparison, not a framework benchmark.

Inspection interfaces. Rakutko's distinct minijev project compares frozen letter-logit scoring with constrained JSON and provides an inspection bench (Rakutko, 2026); our rename avoids name confusion. ChainForge supports visual prompt experiments (Arawjo et al., 2024); LLM Comparator supports response comparisons (Kahng et al., 2025). Langfuse's Jev editor exposes typed questions, distributions, confidence, and requests via external inference (Langfuse, 2026). Phoenix preserves Choice probability metadata (Arize AI and Phoenix contributors, 2026); its GUI display remains unverified. These source-based comparisons establish neither usability nor performance differences. LogitTrail supplies local inspection and auditable evaluation; its live UI lacks cyclic averaging and multi-run comparison. Appendix A separates scopes.

Sensitivity and calibration. Order/token bias and fixed-binding changes are established (Wei et al., 2024; Zheng et al., 2024). Hanna and Feng (2026), a recent preprint, also report that reduced order sensitivity need not improve accuracy; we do not claim that distinction as a new general finding. Our contribution is its concrete examination alongside typed outputs and measured call costs. Temperature scaling (Guo et al., 2017) differs from contextual label-bias correction (Zhao et al., 2021). Canonical key sorting ensures software equivalence under dictionary insertion changes, not model invariance to option meaning or placement. The frozen 35B configuration establishes neither training gains nor superiority over other typed-output systems.

Limitations

Presentation studies cover three checkpoints on one Apple Silicon device; matched parity/latency covers one native quantized model in one session. Architecture, precision, and backend differences prevent model-size attribution. Japanese public subsets have unknown training contamination and source dependence; the English interface example does not establish English task quality. Local data are AI-authored and development-related, without independent human reference validation. No user study establishes usability or downstream decision quality.

Candidate probabilities vary with prompts, labels, coverage, and temperature. Concentrated wrong answers occur; type validity and entropy concentration cannot replace application-specific validation before consequential actions. Score assumes equally spaced stages; Noul is a relative candidate probability, not a verified truth frequency. Questions run sequentially on a large model; other operating systems and lower-memory configurations remain unvalidated. Live-demo timings are not directly comparable with the research harness's controlled interval.

Acknowledgements

AI coding agents accessed through OpenAI Codex assisted under the author's direction with implementation, question authoring, software checks, experiments, analysis, literature organization, and drafting. The independent numerical audit used another AI agent, not independent human review. AI-authored evaluation items have no claimed human-expert validation. The author is responsible for this manuscript and its claims.

Ethics and Availability

The demonstration supports model inspection and typed-interface prototyping, with no evidence for high-stakes deployment. Inference is local; users remain responsible for inputs and downstream use. Code and self-authored local data use MIT terms. External-dataset-derived selection metadata, rubrics, protocols, and results separately retain applicable CC BY-SA 4.0 attribution and share-alike terms. Upstream dataset text, model weights, and compiled binaries are not bundled; repository data cards and notices specify acquisition and attribution.

References

Diogo Almeida. 2026. Introducing system one models & jev. Published 2026-09-15; accessed 2026-09-20.

Ian Arawjo, Chelse Swoopes, Priyan Vaithilingam, Martin Wattenberg, and Elena L. Glassman. 2024. ChainForge: A visual toolkit for prompt engineering and LLM hypothesis testing. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pages 1–18. Association for Computing Machinery.

Arize AI and Phoenix contributors. 2026. Phoenix evaluators: Support for AI SDK evaluation models. Source change included in JavaScript phoenix-evals 2.6.0, released September 22, 2026; accessed September 27, 2026.

dottxt and Outlines contributors. 2026. Outlines: Output types. Official documentation; accessed September 25, 2026.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-constrained decoding for structured NLP tasks without finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

ggml-org. 2026. Qwen3.6-35b-a3b gguf. Pinned Q4_K_M artifact; accessed 2026-09-20.

ggml-org and contributors. 2026. llama.cpp. Pinned revision used by the native helper.

Guidance contributors. 2026. Guidance: A guidance language for controlling large language models. Official repository; accessed September 25, 2026.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330. PMLR.

Karl Hanna and Chen Feng. 2026. Accuracy and order sensitivity diverge under label-free strategies. *arXiv preprint arXiv:2608.11947*. Version 1, August 12, 2026; accessed September 25, 2026.

Minsuk Kahng, Ian Tenney, Mahima Pushkarna, Michael Xieyang Liu, James Wexler, Emily Reif, Krystal Kallarackal, Minsuk Chang, Michael Terry, and Lucas Dixon. 2025. LLM Comparator: Interactive analysis of side-by-side evaluation of large language models. *IEEE Transactions on Visualization and Computer Graphics*, 31(1).

Kentaro Kurihara, Daisuke Kawahara, and Tomohide Shibata. 2022. JGLUE: Japanese general language understanding evaluation. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 2957–2966. European Language Resources Association.

Langfuse. 2026. Jev as a judge. Official documentation for the experimental evaluator; accessed September 27, 2026.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2025. Prompt repetition improves non-reasoning llms. *arXiv preprint arXiv:2512.14982*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

LMQL contributors. 2026. LMQL generations API: Continuation scoring. Official documentation; accessed September 25, 2026.

Mingyu Derek Ma, Yanna Ding, Zijie Huang, Jianxi Gao, Yizhou Sun, and Wei Wang. 2025. Inferring from logits: Exploring best practices for decoding-free generative candidate selection. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33125–33144, Vienna, Austria. Association for Computational Linguistics.

Qwen Team. 2025. Qwen3-reranker-8b: Official model card and inference example. Accessed 2026-09-20; primary source for yes/no candidate-logit extraction.

Qwen Team. 2026. Qwen3.6-35b-a3b: Official model card. Source-model revision 995ad96eacd98c81ed38be0c5b274b04031597b0; accessed 2026-09-20.

Mikhail Rakutko. 2026. mini-Jev: Read the letter. `https://github.com/r-ms/mini-jev`. Software and experimental technical note. Accessed 2026-09-29.

Timo Schick and Hinrich Schütze. 2021. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269. Association for Computational Linguistics.

Taiga Someya, Yushi Sugimoto, and Yohei Oseki. 2024. JCoLA: Japanese corpus of linguistic acceptability. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation*, pages 9477–9488. ELRA and ICCL.

TypeSafe. 2026a. System one adapter for python. Official repository; accessed September 25, 2026.

TypeSafe. 2026b. Typesafe api reference. Accessed 2026-09-20; interface inspiration, not implementation equivalence.

Qihang Wang, Jinwei Tan, Mengyuan Shi, Mayank Sharma, Shuai Zhao, Fuxian Li, Ryan Yan, Alexander P. Kreuzer, Mohit Jain, Dheeraj Toshniwal, and Manoj Seethamsetty. 2026. Single-token expected-value scoring for cold-start candidate ranking. *arXiv preprint arXiv:2609.18188*. Version 1, September 16, 2026.

Sheng-Lun Wei, Cheng-Kuang Wu, Hen-Hsen Huang, and Hsin-Hsi Chen. 2024. Unveiling selection biases: Exploring order and token sensitivity in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 5598–5621, Bangkok, Thailand. Association for Computational Linguistics.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. 2025. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*. Section 2 describes yes/no next-token reranking; inference details are also available in the official Qwen3-Reranker model card.

Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 12697–12706. PMLR.

Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. 2024. Large language models are not robust multiple choice selectors. In *The Twelfth International Conference on Learning Representations*. ICLR 2024 Spotlight; author manuscript version 4, February 22, 2024; conference record https://openreview.net/forum?id=shr9PXz7T0.

Honglei Zhuang, Zhen Qin, Kai Hui, Junru Wu, Le Yan, Xuanhui Wang, and Michael Bendersky. 2024. Beyond yes and no: Improving zero-shot LLM rankers via scoring fine-grained relevance labels. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 358–370, Mexico City, Mexico. Association for Computational Linguistics.

A Comparison Scope and Evidence Access

Comparison	Shared conditions	Recorded observation	Unmeasured or different
Native one-token	Checkpoint, full prefix, candidate IDs/logits, greedy selection	1,350/1,350 labels match; paired latency interval includes zero	Other machines and independent sessions
Grammar JSON	Resident model, selected tasks, HTTP timing interval	Validity, task-specific quality and serialization latency	Prompt and generated length differ; no probability-vector parity
LMQL 0.7.3	Checkpoint, full prefix, candidate IDs; custom native backend	12/12 labels match; 3/12 probability comparisons exceed 10^{-5}	Native evaluation paths differ; not a stock-backend benchmark
ChainForge 0.3.7.4	Same local API, request content and typed-response contract	Eight answer pairs identical; three invalid inputs rejected	Browser flow editor and human performance
Langfuse / Phoenix	Primary-source feature descriptions only	Typed evaluation and probability inspection/metadata precedents	No matched inference, GUI trial or comparative user outcome

Table 4: Evidence tiers, not a capability ranking. Unmeasured properties are not absent capabilities. Integration diagnostics are separate from the three research panels.

Public and internal inspection. The released HTTP response and browser expose semantic labels, typed values, candidate probabilities, concentration, and probability-semantics metadata. Raw candidate token IDs and logits belong to the internal engine and research traces; they are not fields in the public response. Table 4 distinguishes controlled comparisons, integration checks, and source-based related work. These observations do not establish that people work faster or make better decisions with this interface.

Model-free evidence review. The linked repository includes a frozen source/evidence ZIP and a reviewer guide. Its standard-library verification checks the archive and recorded request accounting without weights, network access, or inference. A separate CPU replay executes the six frozen analyses in a new directory and compares all 56 derived files. The three studies contain 26,050 measured requests; 300 earlier pilot predictions are retained as background and excluded from that count. The current manuscript and later framework diagnostics are separate from the frozen ZIP. Replaying recorded data verifies calculations, not the truth of the original observations or their generalization.

Post-hoc LMQL diagnosis. Recomputing the same 12 saved comparisons preserves the three 10^{-5} probability-tolerance failures (`score-01`, `score-03`, `score-04`). Maximum Score expectation error is 7.18×10^{-5}, within the separate, original 10^{-4} typed-value tolerance. All native top-two candidate probability margins exceed 0.9799; matching labels therefore provide no near-boundary test. Candidate softmax over saved LMTP logits reproduces saved LMQL probabilities within 3.64×10^{-9}. Recorded native-path logit differences can account for the observed discrepancies; their underlying computational cause remains unisolated. Type, candidate count, and assistant prefix are confounded, so the failures do not identify a Score-specific defect. This analysis adds no inference calls, changes no threshold, and does not convert the failed probability comparisons into passes.

Human evaluation boundary. No human study was conducted. An AI-operated rehearsal of a separate synthetic fixture page checks task wording, response saving and scoring; it is software/material QA, not an evaluation of the released interface's usability. Its answers and timings are excluded from the experimental results. The present evidence targets system behavior and reproducibility. Human task performance, cross-machine replication, and matched competitor-interface evaluation remain open.