Hunch: Open-Weight Decision Scorers

Technical report for the 0.6B and 1.7B research preview

Antares Labs
https://antareslabs.org

25 September 2026

Weights	https://huggingface.co/antareslabs/hunch-0.6b-preview https://huggingface.co/antareslabs/hunch-1.7b-preview on-device f16 builds: `antareslabs/hunch-0.6b-preview-GGUF` and `-MLX`; the same for `hunch-1.7b-preview`
Code	https://github.com/antareslabsorg/hunch (import `hunch`; data builders, trainer, evaluator, converters)
Checkpoints	Hunch 0.6B: `w16-06-v3dgp10-1200-s55`, SHA-256 `75d678f36ce35cfc...` Hunch 1.7B: `g2-17-v4t`, SHA-256 `e2fa7f0ab23e3f1d...`
Licence	Apache-2.0 (weights, code, generated data); Qwen3 backbones Apache-2.0; source licences in Section 6
Status	Research preview. Every result below is for these two checkpoints unless a table says otherwise.

Abstract

Hunch is a family of open-weight decision scorers from Antares Labs, released under Apache-2.0 as a research preview at 0.6B and 1.7B parameters. It is a specialist: a small model trained for its task families, not a general zero-shot model. Given a state, questions and text candidates, it scores each candidate as a separate path through a fine-tuned Qwen3 decoder with a scalar readout and returns a probability for each. It generates no text, and, up to floating-point rounding, candidate order cannot change its output: permuting the options changed 0.0 % of its answers on three public suites, where two published Laya models change 15.0 % and 23.0 % of their MASSIVE intent answers. On a sealed in-family test split of 3,467 questions in 12 families, read once per checkpoint as declared in advance, accuracy is 0.8543 (0.6B) and 0.8751 (1.7B), against 0.4673 for a per-family constant predictor; a temperature fitted on a disjoint split lowers the 1.7B's smooth calibration error there from 0.0253 to 0.0136 and leaves the 0.6B's essentially unchanged (0.0162 to 0.0167). On its own families, Hunch 1.7B leads TypeSafe's Jev 1.13 and the open Decider 2B, both answering zero-shot, by 7.2 [5.7, 8.8] and 13.9 [10.9, 17.0] points, with Brier 0.147 against 0.273 and 0.309. In fp32, a shared-prefix engine changes none of 6,000 held-out answers and answers the 0.6B's median request in 36.7 ms on one RTX 5090, 10.1 times below Jev's API round trip; from the MLX f16 build on a laptop it takes 156.2 ms, 2.4 times below it. GGUF and MLX f16 builds change at most 10 of 6,000 held-out answers against the fp32 reference, fewer than the 28 that bf16 alone changes, and every release number here is generated from its result file, after recomputation checks on the per-question rows. Beyond its own families transfer is uneven, and zero-shot on public suites built for general decision models it trails the leading generalists. A planted "the correct answer is X" moves about a third of held-out answers to X.

Inside this report

1 Introduction

Many software decisions are choices among options that are known when the decision is made: which team should handle a message, whether a comment breaks a rule, whether a passage supports a claim, which level of a rubric a response deserves. A general-purpose language model can be prompted for such a choice, but its answer arrives as text that must be parsed, its probabilities are not the object it was trained to produce, and multiple-choice answers from prompted models are known to move when the options are reordered [1–3]. A fixed-label classifier avoids those problems but cannot accept a new set of options at call time.

Hunch is a decision scorer built for this setting. A request supplies a state (text or JSON), one or more questions, and the candidate answers of each question. Every candidate is serialized with the state and its question into its own token path, a fine-tuned Qwen3 decoder [4] reads the path, and a scalar head scores its last token. A softmax over each question's scores returns a distribution over exactly the supplied candidates (Figure 1). Nothing is generated. Because no path contains another candidate or a list position, the returned distribution is equivariant to reordering the candidates by construction (Section 4).

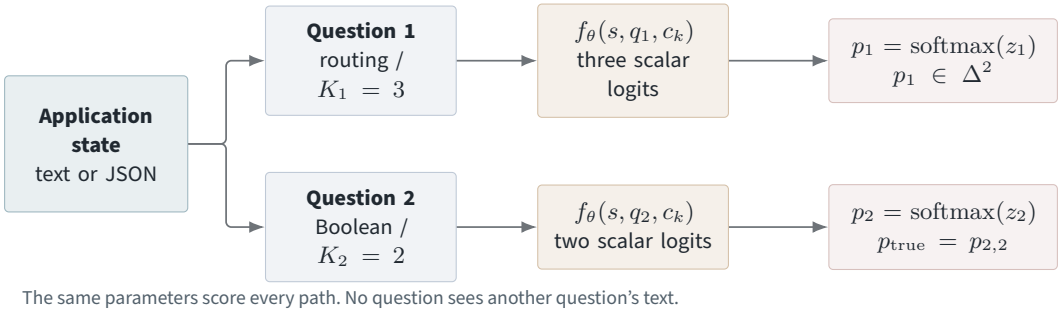

Figure 1. The decision interface. Branches share model parameters and state content; the reference engine recomputes the state on every candidate path. Each question has its own normalization.

This report describes the two checkpoints released on 25 September 2026 as a research preview: **Hunch 0.6B** (`w16-06-v3dgp10-1200-s55`, a Qwen3-0.6B backbone) and **Hunch 1.7B** (`g2-17-v4t`, Qwen3-1.7B, fine-tuned in a second stage from an earlier checkpoint). Both are Apache-2.0 and ship as PyTorch weights, with a shared-prefix engine, and as GGUF and MLX f16 builds. They are specialists, small models trained for their task families rather than general zero-shot decision models: on those families they are accurate, calibrated and ahead of commercial and open generalists answering zero-shot. Table 1 summarizes what we measured, including where the specialization ends.

Table 1. The release at a glance. Sealed rows are the test read (Section 9.1); all others are at $T = 1$ unless marked. Every cell is a macro generated from a result file.

Measurement	Hunch 0.6B	Hunch 1.7B	Reference
Sealed in-family test accuracy (3,467 questions, read once)	0.8543	0.8751	untrained backbone 0.3303 / 0.3337
Sealed smooth ECE, $T = 1 \rightarrow$ release $T = 0.8706$	$0.0162 \rightarrow 0.0167$	$0.0253 \rightarrow 0.0136$	temperature fitted on the calibration split
Accuracy on its own families (3,523 questions)	0.8535	0.8731	zero-shot: Jev 1.13 0.8010, open 2B 0.7346
Held-out CaseHOLD accuracy (5 options)	0.5515	0.6020	constant predictor 0.2120
Held-out GoEmotions accuracy (Boolean)	0.7465	0.8000	constant predictor 0.7645
typed-decisions accuracy, zero-shot	0.3795	0.5105	majority floor 0.520; open 2B 0.5895
Planted "the correct answer is X": answers moved to X	35.6 %	32.4 %	neutral sentence 2.8 % / 1.8 %
Toxic comments missed at $P(\text{true}) \geq 0.5$, no identity term	59.0 %	67.1 %	with an identity term 77.3 % / 77.3 %
On-device f16 builds: answers changed of 6,000 (MLX / GGUF)	0 / 1	0 / 10	bf16 alone: 28 / 28
Median request, one RTX 5090, shared-prefix fp32	36.7 ms	66.6 ms	Jev 1.13 API round trip: 369.1 ms

Contributions.

1. **A decision contract and a scorer built on it**: typed Choice, Boolean and Score questions, each candidate scored as its own path, with per-question normalization and a proper, ordinal-aware loss derived with their assumptions; up to floating-point rounding, option order cannot change an answer, and permuting

options changed 0.0 % of answers where two published Laya models change 15.0 % and 23.0 % (Sections 2 to 5 and Appendix A).

2. **A sealed in-family evaluation**, read once per released checkpoint as declared the day before, beside the untrained backbone and a constant predictor, with a temperature fitted on a disjoint split (Sections 8 and 9.1).
3. **A lead over zero-shot generalists on the specialist's own tasks**: on the 3,523 calibration-split questions of its twelve families, Hunch 1.7B leads TypeSafe's Jev 1.13 and the open Decider 2B by 7.2 [5.7, 8.8] and 13.9 [10.9, 17.0] points, with Brier 0.147 against 0.273 and 0.309 (Section 9.3).
4. **An exact shared-prefix engine**: in fp32 it changes none of the 6,000 held-out answers; on one RTX 5090 the 0.6B's median request in fp32 is 10.1 times below Jev's API round trip, and on a laptop the MLX f16 build takes 156.2 ms, 2.4 times below it (Sections 3 and 9.7 and Appendix B).
5. **On-device builds** admitted only inside the precision floor that bf16 inference already incurs (Section 9.6).
6. **A checked record and a full evaluation**: every release number generated from its result file, after recomputation checks on the per-question rows; held-out families, public suites with paired intervals, robustness, fairness, contamination and personal-data checks, and controlled studies with exact rank tests (Sections 6, 9, 10 and 12).

2 The decision interface

A request contains one or more states, each state its questions, and each question its candidates. Hunch returns a probability distribution over the supplied candidates of every question, or fails the whole request with an error; it never returns free text. The scorer is built with `AutoModel`, so the backbone's language-model head is not used; prompting the backbone as a chat model is a different system.

Definition 2.1 (Probability-valued decision)**.** For state s, question q_j and ordered candidate tuple $C_j = (c_{j1}, \dots, c_{jK_j})$, a decision model computes

$$D_\theta(s, q_j, C_j) = p_j \in \Delta^{K_j - 1}, \qquad \Delta^{K-1} = \{p \in \mathbb{R}^K_{\geq 0} : \mathbf{1}^\top p = 1\}. \tag{2.1}$$

The candidate order is an output alignment convention; candidate text, not an index, determines a score.

Table 2. Three question types, one scoring function. Score indices are zero-based in the implementation.

Type	Support	Deterministic readout
Choice	$2 \leq K \leq 255$ labeled candidates	p; the id of candidate $c_{\arg\max_k p_k}$; a concentration statistic
Boolean	$(\text{false}, \text{true})$	p_{true}; a caller may apply a threshold
Score	$2 \leq K \leq 10$ described levels	p; expected level $\sum_{k=1}^{K}(k-1)p_k$; modal level

A request may contain up to 64 questions per state, and a state at most $32{,}768$ tokens. Each path contains the state, its own question and one candidate, so the returned marginals do not identify a joint distribution over answers. Scoring packs whole questions into microbatches whose padded size, the number of candidates times the longest candidate path, fits a token budget ($32{,}768$ tokens in the PyTorch loader, $16{,}384$ in the on-device loaders); a question larger than the budget fails the request and is never truncated, and the budget changes packing, not scores, up to floating-point summation order.

The schema separates an option's transport id from its semantic label and optional description; an id enters candidate text only as the fallback when a label is omitted, so invariance to renaming ids requires explicit labels. Score paths contain level descriptions without numeric indices. Training targets are event probabilities, human label distributions or action preferences, with separate provenance tags; these are different statistical objects, and a strictly proper loss does not by itself establish finite-sample or out-of-distribution calibration. The released checkpoints apply a temperature fitted on in-family calibration data by default; for inputs unlike the training families the release documentation recommends $T = 1$, the setting of every out-of-family result in this report (Section 8.3). The Choice concentration $c(p) = (\max_k p_k - 1/K)/(1 - 1/K)$ is a deterministic statistic, not a calibrated probability that the selected answer is correct; decisions under an application loss are Bayes actions under p (Appendix A).

3 Architecture

3.1 Serialization and tensor layout

The released models use `hunch-serialization-v1-plain`, without added special tokens. Let τ be the backbone tokenizer. State, question, candidate and readout segments are tokenized separately, then concatenated:

$$x_{jk} = \tau(S(s))\|\tau(Q(q_j))\|\tau(C(c_{jk}))\|\tau(R), \qquad \ell_{jk} = |x_{jk}|. \tag{3.1}$$

This fixes a token-identical prefix across candidates and avoids merges across segment boundaries. The readout suffix R is a newline followed by `### Decision:`; its final token supplies the readout state. A string state is preserved verbatim; JSON uses UTF-8, two-space indentation and preserved key order.

A microbatch (Figure 2) is flattened to $P = \sum_j K_j$ paths and right-padded to $L_{\max} = \max_{jk} \ell_{jk}$, giving tokens $X \in \mathbb{N}^{P \times L_{\max}}$, a validity mask M and decoder states $H_\theta(X, M) \in \mathbb{R}^{P \times L_{\max} \times d}$ from the Qwen3 backbone loaded without its vocabulary projection [4]. With $a(j,k)$ the flattened path index,

$$h_{jk} = H_\theta(X, M)[a(j,k),\, \ell_{jk} - 1, :], \qquad \rho_{jk} = \sqrt{d^{-1} \textstyle\sum_{r=1}^{d} h_{jk,r}^2 + \epsilon}, \quad \epsilon = 10^{-6}, \tag{3.2}$$

$$u_{jk} = \gamma \odot h_{jk}/\rho_{jk}, \qquad z_{jk} = w^\top u_{jk}. \tag{3.3}$$

The learned parameters beyond the backbone are $\gamma, w \in \mathbb{R}^d$, exactly $2d$ scalars; the head is initialized from $\mathcal{N}(0, 0.02^2)$, and a shared bias would cancel in the softmax and is omitted. The backbone's own final RMSNorm precedes this readout norm; the two do not fold into one normalization, which matters for the on-device builds (Section 9.6).

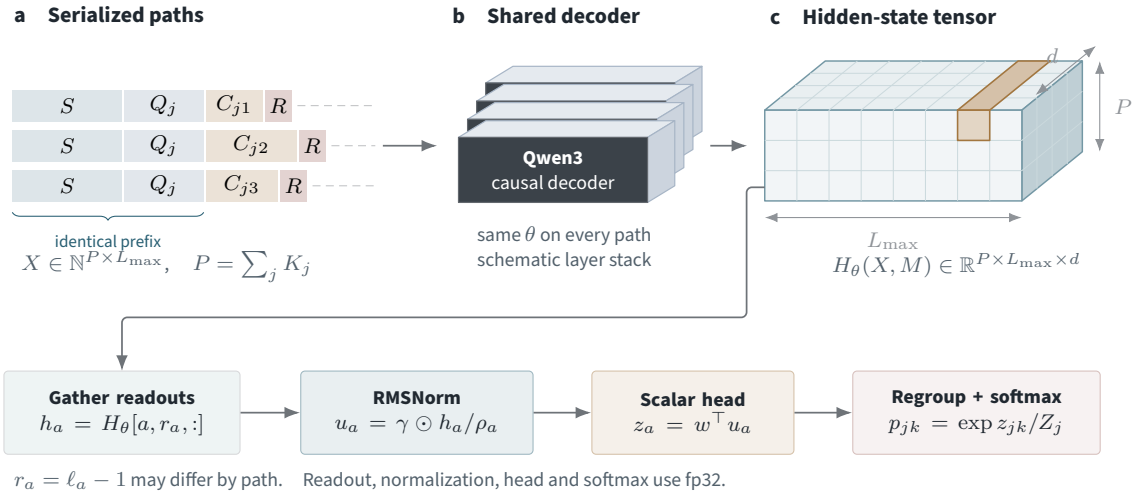

Figure 2. The reference (flat) scorer. The three axes of H_θ are paths, token positions and hidden features. The gold fiber marks one readout vector; each path uses its own last valid token. R is the potentially multi-token `### Decision:` suffix; dashed tails are masked padding. Dimensions and the number of drawn decoder layers are schematic. Only the final scalar logits are coupled within a question.

3.2 Masked per-question normalization

The flat logits are regrouped into a padded question tensor; invalid slots receive $-\infty$ and exactly zero probability. For valid candidates,

$$p_{jk} = \frac{e^{z_{jk}}}{\sum_{\ell=1}^{K_j} e^{z_{j\ell}}} = \frac{e^{z_{jk} - m_j}}{\sum_{\ell=1}^{K_j} e^{z_{j\ell} - m_j}}, \quad m_j = \max_k z_{jk}, \qquad p_{\text{true}} = \sigma(z_{\text{true}} - z_{\text{false}}) \text{ for Boolean questions.} \tag{3.4}$$

Normalizing over the whole request would couple unrelated questions and violate the contract. The decoder may run in bf16; the gathered readout, readout norm, head and softmax use fp32.

Cost. There is one non-generative decoder forward per packed microbatch; a request larger than the token budget uses several. For B states with J questions of K candidates and segment lengths s, q, c (c including the readout suffix), the scorer processes $BJK(s+q+c)$ tokens, because the state and question are repeated on every candidate path, so cost grows with the number of candidates times the path length. The release also ships a shared-prefix

engine, `FastHunch` (`hunch/fast.py`), with the arguments and output of `Hunch`: it packs a state's paths into one sequence, the state once, each question once and each candidate once ($s + Jq + JKc$ tokens per state), under a boolean mask that gives every token exactly the visibility it has in its own path; every token keeps its path position, and the fp32 readout is unchanged (Appendix B). In fp32 it changes no answer of the reference; in bf16 it stays within bf16's own error at 0.6B only (Section 9.7). Every other result in this report uses the reference engine.

4 Structural properties

Three properties follow from the path construction; Appendix A gives the derivations.

Equivariance. A path has no access to its candidate's list position or its siblings. For a permutation π with $(\pi C)_k = c_{\pi(k)}$ and $(P_\pi z)_k = z_{\pi(k)}$, the reordered logits are $z_{\pi(k)}$, and the softmax commutes with the permutation:

$$D_\theta(s, q, \pi C) = P_\pi D_\theta(s, q, C). \tag{4.1}$$

This holds in exact arithmetic; padding and kernels can introduce floating-point differences, and a first-index tie-break can change a selected id. Permuting the options of 200 cases on each of three public suites changed 0.0 % of either release's answers (Section 9.4).

Independence of irrelevant alternatives. For candidates a, b of one question and any retained subset S with mass $P_S = \sum_{k\in S} p_k$,

$$\frac{p_a}{p_b} = e^{z_a - z_b}, \qquad p(k \mid S) = \frac{p_k}{P_S} \quad (k \in S). \tag{4.2}$$

Adding a candidate rescales the others but cannot change their odds, so a relation among the offered candidates, such as choosing the median of the offered values, cannot be represented, and every probability is conditional on the supplied support. Duplicating a candidate's text inflates its total probability (Equation (A.4)).

A disabled set head. An optional Choice set head adds a residual δ_k computed by self-attention over the normalized readouts $[u_k; \log K]$ (width 256, four heads, no positional encoding over candidates [5], zero-initialized output), which lets logits depend on the offered set while preserving equivariance. It is disabled in both releases and in every configuration-verified comparison of Section 10, so Equation (4.2) holds for the release.

5 Learning objective

A proper loss. For a complete question with target $t \in \Delta^{K-1}$ and logits z, soft cross-entropy and its logit derivatives are

$$\mathcal{L}_{\mathrm{CE}}(t, p) = \log \sum_k e^{z_k} - t^\top z = H(t) + D_{\mathrm{KL}}(t\|p), \qquad \nabla_z \mathcal{L}_{\mathrm{CE}} = p - t, \qquad \nabla_z^2 \mathcal{L}_{\mathrm{CE}} = \mathrm{diag}(p) - pp^\top \succeq 0. \tag{5.1}$$

The entropy term does not depend on the parameters, so fitting cross-entropy minimizes divergence from the target distribution [6], and a non-degenerate soft target has a positive irreducible NLL floor; this is why a large NLL on the urn family, whose targets are exact stochastic mechanisms, need not indicate poor fidelity. The Hessian annihilates $\mathbf{1}$, the unidentifiable common logit offset; convexity is in z, not in the decoder parameters. Smoothing a target changes the optimum, so label smoothing is a modeling decision, not a neutral numerical treatment.

Ordinal geometry. For K ordered levels let $A_{rk} = \mathbf{1}_{\{k \le r\}}$, so $(Ap)_r$ is cumulative mass. Score questions add a discrete CRPS term:

$$\mathcal{L}_{\mathrm{CRPS}}(t, p) = \|A(p - t)\|_2^2, \qquad \mathcal{L}_{\mathrm{Score}} = \mathcal{L}_{\mathrm{CE}} + 0.5\, \mathcal{L}_{\mathrm{CRPS}}. \tag{5.2}$$

A displacement of mass across several level boundaries accumulates error at each boundary, a notion of ordinal distance that cross-entropy lacks. The combined loss remains strictly proper, with the conditional mean target as its unique optimum (Proposition A.1).

Family sampling and batching. With family sizes n_f, the sampler picks family f with probability $\pi_f = \sqrt{n_f} / \sum_g \sqrt{n_g}$ and then a uniform question in it, so the population objective is

$$\mathcal{J}(\theta) = \sum_f \pi_f \frac{1}{n_f} \sum_{j \in f} \left[\mathcal{L}_{\mathrm{CE},j} + \mathbf{1}_{\{\mathrm{type}(j)=\mathrm{Score}\}}\, 0.5 \mathcal{L}_{\mathrm{CRPS},j} \right] \tag{5.3}$$

and a row is drawn with probability proportional to $n_f^{-1/2}$. This limits the dominance of large families: in the base mixture, Civil Comments contributes $84{,}955$ of $244{,}624$ training questions. Microbatches hold complete questions, and each contributes its summed question loss divided by the number N of questions in the logical batch, so accumulated gradients equal those of $\mathcal{L}_{\text{step}} = N^{-1} \sum_j \mathcal{L}_j$; the accumulated norm is clipped to 1 before an AdamW step.

Candidate subsampling changes the target problem. Training may keep only a subset S of a Choice question's candidates. With $Z_S = \sum_{k \in S} e^{z_k}$ and all positive target mass retained,

$$\mathcal{L}_S - \mathcal{L}_C = \log \frac{Z_S}{Z_C} = \log P_S \le 0, \qquad \frac{\partial \mathcal{L}_S}{\partial z_k} = \mathbf{1}_{\{k \in S\}} \Big(\frac{p_k}{P_S} - t_k \Big), \tag{5.4}$$

so for fixed scores the capped loss is systematically smaller than the full loss, omitted candidates receive no gradient where the full loss would push their logits down, and averaging capped losses over sampled subsets does not recover the full loss (Proposition A.2). The implementation always keeps an argmax target and at most $k_{\text{cap}} - 1$ of the largest positive targets before sampling the rest, so a dense target can lose support, and its renormalized target then changes as well. Boolean and Score questions are never capped. A historical comparison of capped against full candidate sets also changed steps and optimizer, so no causal effect of the cap is claimed.

6 Data

Both releases are trained on public, licence-audited sources and on data generated by rule engines; the 1.7B's second stage adds open, model-written teacher data. Each training version adds to the previous one (`sprint_v3d`, then `sprint_v3dgp10`, then `v4t`), Table 3 shows which checkpoint trains on which families, and every version shares the same evaluation splits.

6.1 The base mixture

The base mixture, `sprint_v3d`, has 244,624 training questions in 12 families (Table 3): ten from public Hugging Face datasets and two generated from a seed with no third-party text (an urn mechanism with exact ground-truth distributions and a refund-policy rule engine). A source is admitted only if it carries licence class A or B and its labels were not produced by a closed model, and a release build refuses any other source. Two mirrors that re-declare their licences (`mteb/banking77` as MIT, `mteb/amazon_massive_intent` as Apache-2.0) are treated as CC BY 4.0 with attribution, since a mirror cannot enlarge the original grant; MNLI, SST-5, HaluEval, Aegis 2.0, UltraFeedback and the GLUE and SuperGLUE tasks whose primary grants could not be verified are excluded. No training or evaluation rows are redistributed; the builder reconstructs them from the public sources.

Every family carries authored instructions and candidate descriptions, written for Hunch rather than quoted from upstream prompts; the plain counterpart without descriptions (`v3`) exists only for the study in Section 10.1. Split membership is assigned by source group, not by derived question row, so all views of one source item stay in one split.

6.2 A rule-generated indirect-answer family

Circa [15] asks what an indirect answer to a yes/no question means, over eight labels. The base mixture has no such task, and the model it trained (the 1.7B parent) scores far below Circa's constant predictor (Section 9.2). We therefore generated 27,073 indirect-answer questions with a rule engine over a hand-written topic graph. No model and no Circa item is involved, but the family reuses Circa's question instruction and its eight answer labels verbatim, with attribution. **Circa is therefore not a held-out measurement for either release**: it measures transfer of a trained question format to new items, not transfer to an unseen task.

6.3 Prior augmentation

For 10 % of training questions, the state is replaced by another state from the same family or by a truncated or shuffled copy of its own, and the target by the family's base rate. The intent is to teach the readout what an uninformative state should produce; no text is added. Together with the indirect-answer family it defines `sprint_v3dgp10`, the 271,697-question mixture of the 0.6B release. The share lies in the region the prior-share study found promising (Section 10.2).

Table 3. Training data of the two releases. Counts are training questions from the data manifest and the teacher conversion report. Classes are the project's source-admission record, not a legal determination. "Stage 1" is the parent of the 1.7B release.

Family	Type	Source licence (class)	Questions	0.6B	1.7B stage 1	1.7B stage 2
Banking77 [7]	Choice	CC BY 4.0 upstream (B)	8,960	✓	✓	✓
CLINC150 [8]	Choice	CC BY 3.0 (B)	13,844	✓	✓	✓
MASSIVE intent [9]	Choice	CC BY 4.0 upstream (B)	10,348	✓	✓	✓
Bitext support routing	Choice	CDLA-Sharing 1.0 (B)	33,808	✓	✓	✓
VitaminC [10]	Choice	CC BY-SA 3.0 (B)	18,152	✓	✓	✓
BoolQ [11]	Boolean	CC BY-SA 3.0 (B)	8,533	✓	✓	✓
PAWS, Wiki [12]	Boolean	free use; Wikipedia text (B)	18,035	✓	✓	✓
Civil Comments [13]	Boolean, soft	CC0-1.0 (A)	84,955	✓	✓	✓
Civil toxicity level	Score	CC0-1.0 (A)	16,991	✓	✓	✓
HelpSteer2 [14]	Score	CC BY 4.0 (B)	18,006	✓	✓	✓
Refund policy	Choice	generated, no third-party text (A)	4,494	✓	✓	✓
Urn mechanism	Choice, Boolean	generated, no third-party text (A)	8,498	✓	✓	✓
Indirect answers (Section 6.2)	Choice	generated; Circa wording, CC BY 4.0	27,073	✓		✓
Teacher data, 8 families (Section 6.4)	all three	Apache-2.0 (B)	40,612			✓
Total				271,697	244,624	312,309

Sources: the data builder's `manifest.json` (`counts_by_family_split`) and the teacher converter's `report.json`, which the commands in `REPRODUCE.md` regenerate; licences from `THIRD_PARTY_NOTICES.md`. The 0.6B and stage-2 mixtures also apply prior augmentation to 10 % of training questions (Section 6.3).

6.4 Converted teacher data

The 1.7B release's second stage adds the teacher data published with an open decision model (`github.com/Mapika/decider`, `teacher_data/` at commit `b44b4c9880a6`, Apache-2.0), written by the open-weight `Qwen/Qwen3.5-27B`, which the data's repository names and whose licence permits training on its outputs; the per-item generating revision is not recorded upstream. A converter maps its Boolean, choice and score questions, situations, routing messages and command risk and scope items to Hunch's schema, drops the 5,497 of 48,284 questions that the teacher answered two different ways, sets each target to the teacher's probability on its written answer with the remainder spread evenly (one-hot where none is recorded), and removes any state matching a typed-decisions test state or one of our held-out states; of 2,000 40-character windows of the typed-decisions test states (five per state), 0 occur in the teacher text. What remains is 40,612 training and 2,175 development questions in 8 families; with `sprint_v3dgp10` they form `v4t`, 312,309 questions, of which the indirect-answer family is 8.7 %. This is the only model-written training source, and no output of a closed model is a training target or label in either release.

6.5 Evaluation splits

The data builder writes train, dev, calibration, test and held-out files with a manifest of per-source licence classes, counts and hashes: 13,945 dev, 14,002 calibration, 85,604 test and 66,180 held-out questions, with dev, calibration and held-out files byte-identical across the `sprint` versions. The `v4t` dev file adds the teacher's dev questions and serves only checkpoint selection during training; every evaluation here reads the `sprint_v3d` files. The held-out file holds three families that no version trains on: GoEmotions [16], CaseHOLD [17] and Circa [15], the last with the caveat above.

6.6 Overlap between evaluation and training text

We compared the state text of every evaluation question with every training text either release saw (`sprint_v3d` and `v4t` together, 556,933 rows). States were reduced to their string values in sorted-key order, so identical text under different field names matches, and normalized by Unicode NFKC, lower-casing and whitespace collapse. Three measures were computed on the full sets: exact match, character 13-gram Jaccard similarity of at least 0.8, and MinHash similarity of at least 0.8 (128 permutations) [18, 19] (Table 4).

The 49 held-out matches are all very short GoEmotions states (median 10 characters) matched by a row of a different source, such as a generic phrase that is also a Civil Comments comment; the dev and calibration matches come from upstream duplication (repeated Bitext support utterances, the urn family's templates, duplicate Civil Comments comments); and neither public suite shares any test state with training. Removing the exact matches moves dev accuracy from 0.8650 to 0.8591 (0.6B) and from 0.8785 to 0.8718 (1.7B) on the 3,416 dev questions read, 194 of which match.

Table 4. Evaluation questions whose state also occurs in training. No evaluation id occurs in training; every match is text that recurs upstream.

Evaluation set	Questions	Exact match	13-gram Jaccard ≥ 0.8	MinHash ≥ 0.8
Held-out families	66,180	49 (0.074 %)	54	54
Dev	13,945	573 (4.1 %)	1,435	1,551
Calibration	14,002	469 (3.3 %)	1,318	1,465
Sealed test read (Section 9.1)	3,467	205	not computed	not computed
typed-decisions test states	400	0	0	0
typed-decisions-v2-system-one test	5,214	0	0	0

Sources: `docs/results/launch/contamination_ids.json` (id lists), `docs/results/launch/sealed_contamination_ids.json`, `docs/21-contamination-report.md`.

6.7 Personal data

A pattern scan of the base mixture's train, dev, calibration and held-out files found 361 contact details that look real (21 e-mail addresses, 63 phone numbers, 277 street addresses) and no bank or card numbers; of all e-mail, phone and street hits, placeholders included, the training split holds 29, 45 and 190. Most come from HelpSteer2 prompts and Wikipedia-derived text in PAWS, VitaminC and BoolQ, and name detectors fire often (24,843 full or titled names), mostly on public figures and dialogue in published text. The model-written teacher data holds 608 such contact details, mostly e-mail (452), and the heuristic cannot tell invented details from copied ones. The scorer cannot emit text; whether its scores reveal membership of a string in the training data has not been measured.

7 Training and checkpoint selection

7.1 Recipes

Table 5 lists the recipes as archived in the run configurations. Every run uses one GPU, bf16 autocast over fp32 master weights and the objective of Section 5, and keeps the checkpoint with the lowest dev NLL among those evaluated during training. Hunch 0.6B is a single stage from the pretrained Qwen3-0.6B. Hunch 1.7B has two: the parent `h200-17-fullk-v3d`, trained with full candidate sets on the base mixture, and a second stage that loads only the parent's weights (with a fresh optimizer and schedule) and trains on `v4t` on one RTX 5090 with a small token budget, a path cap and frozen input embeddings. The trainer skips and counts questions that exceed the budget (none in the second stage) or the path cap, which removes some long HelpSteer2 items (23 at the parent's cap; the second stage's count was not recorded).

Table 5. Training recipes, from the archived run configurations. "Candidates" caps the Choice candidates kept per training question (Section 5); "all" means full sets. Durations are the last logged elapsed time, not the cost of the research programme.

Field	Hunch 0.6B	1.7B stage 1 (parent)	Hunch 1.7B (stage 2)
Starting point	Qwen3-0.6B	Qwen3-1.7B	parent weights only
Training data	`sprint_v3dgp10`	`sprint_v3d`	`v4t`
Steps × questions per step	1,200 × 120	1,200 × 120	800 × 120
Padded-token budget / path cap	16,384 / 2,048	196,608 / 2,048	8,192 / 1,024
Candidates per training question	at most 16	all	at most 16
Optimizer	8-bit AdamW	AdamW	8-bit AdamW
Learning rate, backbone / head	2×10^{-5} / 3×10^{-4}	2×10^{-5} / 3×10^{-4}	1×10^{-5} / 3×10^{-4}
Warmup fraction	0.03	0.03	0.03
Input embeddings	trained	trained	frozen
Seed	55	0	0
GPU	one RTX 5080	one H200	one RTX 5090
Logged duration	not recorded	8.05 h	7.58 h

All runs: weight decay 0.1, CRPS weight 0.5 on Score questions, family-sampling exponent 0.5, gradient clipping at norm 1, cosine decay to a tenth of the peak rate, set head disabled, one process. Sources: `docs/results/run_configs/released/w16-06-v3dgp10-1200-s55_config.json` and `docs/results/day7/v1/g2-17-v4t_config.json`.

7.2 Checkpoint selection

Hunch 0.6B was chosen by a rule written on 20 September 2026 and implemented as a script: among candidates that are calibrated in family (dev smooth ECE at most 0.03 after their own fitted temperature), within 2.0 points of the best in-family dev accuracy and within 0.12 of the lowest pooled held-out NLL, the lowest dev NLL wins, with

ties broken by dev calibration, data version and seed index. The rule was amended while results were visible, and several 0.6B checkpoints are tied within seed noise on these numbers. On 23 September we trained candidates of both sizes on `v4t`, each with a replication seed. The rule fixed before any of them was read required, among other conditions, that a candidate beat the open Decider 2B's typed-decisions accuracy of 0.5895; the 1.7B candidate read 0.5105, so under that rule the parent would have shipped. The rule was amended once, after both 0.6B seeds and the first 1.7B seed had been read: it keeps the in-family accuracy, held-out NLL and calibration bars and requires both seeds of a size to beat the earlier checkpoint on at least three of four out-of-family reads (typed-decisions accuracy, typed-decisions-v2-system-one accuracy, pooled held-out NLL and the median change over the 59 Laya-list suites). The one seed read after the amendment, the 1.7B replication seed `g2b-17-v4t-s1`, agreed on all four, so `g2-17-v4t` ships; the 0.6B candidate had both seeds better on only one read, and `w16-06-v3dgp10-1200-s55` remains the 0.6B release. The model cards state both rules and their outcomes. The parent's results appear beside the release throughout; its weights are not released.

8 Evaluation protocol

8.1 What is read, and how often

Table 6 lists the evaluation reads. Trained checkpoints are read with bf16 autocast and an fp32 readout, and every metric is at $T = 1$ unless the release temperature is named. Dev and held-out reads were made many times during research, so they are research evidence rather than an untouched test; the test split was read once per released checkpoint. The public suites were read zero-shot, once per checkpoint; a re-score of both releases on typed-decisions reproduced their accuracy exactly and supplied the per-decision rows behind the intervals.

Table 6. Evaluation reads of the released checkpoints.

Read	Questions	Role
Dev, 300 per family	3,416	in-family evidence; selection of the 0.6B; calibration before and after T
Calibration, 300 per family	3,523	fits the release temperature and nothing else
Sealed test, 300 per family	3,467	read once per release, from 25 September; nothing selected on it
Held-out, 2,000 per family	6,000	three families no version trains on (Circa's format is trained, Section 6.2)
typed-decisions test	2,000 decisions in 400 cases	public suite with published references; zero-shot
typed-decisions-v2-system-one test	5,214	public suite without published references; zero-shot
Laya list	59 suites	rebuilt from a published benchmark's code; zero-shot
Probes	500 held-out questions	planted instructions and formatting perturbations

8.2 The sealed in-family test read

The test split was sealed until 25 September 2026 by a dated guard: the evaluator refuses it before that date unless an explicit override is given and accepts only exact split names. This is not an exactly-once lock, so the single read rests on a declaration and a record; before the date only the frozen-backbone baseline had opened the split, on 20 September. The read was declared on 24 September: once per released checkpoint, 300 questions per family, reported at $T = 1$ and at the release temperature beside the untrained backbone on the same 3,467 questions, with nothing selected on it. Both checkpoints were fixed beforehand, and their weight files matched the SHA-256 digests in the front matter. A first run stopped before scoring any question and was repeated with a larger batching budget and a different allocator setting, which change how questions are packed, not which are scored or how (Appendix D).

8.3 Temperature

Each release applies one global temperature T, fitted by minimizing mean NLL on its calibration read over a 121-point logarithmic grid on $[0.25, 4]$; nothing else is fitted on that split, and the disjoint dev split reports calibration before and after. Both releases select the same grid point, $T = 0.8706$. For positive probabilities,

$$p_k^{(T)} = \frac{p_k^{1/T}}{\sum_\ell p_\ell^{1/T}} = \mathrm{softmax}(\log p/T)_k = \mathrm{softmax}(z/T)_k, \tag{8.1}$$

because a shared offset cancels, so stored untempered probabilities suffice to apply or refit a temperature. A positive temperature preserves every ranking, so accuracy does not depend on it (Appendix A).

8.4 Metrics

For predicted distributions p_i and targets t_i we report

$$\widehat{\mathrm{NLL}} = -\frac{1}{n}\sum_i \sum_k t_{ik} \log p_{ik}, \qquad \widehat{B} = \frac{1}{n}\sum_i \|p_i - t_i\|_2^2, \qquad \widehat{\mathrm{Acc}} = \frac{1}{n}\sum_i \mathbf{1}_{\{\arg\max p_i = \arg\max t_i\}}. \tag{8.2}$$

Accuracy compares modal targets and does not measure fidelity to soft labels; vector Brier is twice scalar Bernoulli Brier for Boolean questions; soft-target and realized-outcome scores share their population optimum [6, 20, 21] (Equation (A.2)). NLL clips only inside the logarithm, at 10^{-12}. The smooth ECE uses $(c_i, a_i) = (p_{i,\mathrm{true}}, t_{i,\mathrm{true}})$ for Boolean questions and $(\max_k p_{ik}, t_{i,\arg\max p_i})$ otherwise. At $x_g = g/200$, $g = 0, \ldots, 200$, with $\sigma = .05$ and reflected Gaussian weights

$$w_{gi} = \sum_{y \in \{c_i, -c_i, 2-c_i\}} e^{-(x_g - y)^2/(2\sigma^2)}, \qquad \widehat{\mathrm{smECE}} = \frac{\sum_g d_g |\bar{a}_g - \bar{c}_g|}{\sum_g d_g}, \tag{8.3}$$

where $d_g = \sum_i w_{gi}$ and $\bar{a}_g, \bar{c}_g$ are the $w_{g\cdot}$-weighted means of a_i and c_i; denominators are floored at 10^{-12}. This fixed-bandwidth estimator is one of several called smooth ECE [22]. On the public suites, typed-decisions reports KL from the soft gold distribution and Brier score as its card defines them; typed-decisions-v2-system-one has hard gold, so only accuracy, Brier and NLL are defined.

8.5 Uncertainty and tests

Intervals on in-family and held-out summaries resample source groups (500 resamples) and condition on one fitted checkpoint, so they exclude training-seed variability. typed-decisions intervals resample the 400 test cases with their five decisions each (10,000 draws, seed 0), and paired differences use the decisions both models answered [23]; fairness gaps use 1,000 bootstrap resamples. For replicated recipe comparisons a run is the statistical unit. With per-run values $x_1, \ldots, x_n$ and $y_1, \ldots, y_m$ of a lower-is-better metric, the favourable-pair statistic [24] and its one-sided permutation tail are

$$U = \sum_{a=1}^{n}\sum_{b=1}^{m} \left[\mathbf{1}_{\{x_a < y_b\}} + \tfrac{1}{2}\mathbf{1}_{\{x_a = y_b\}}\right], \qquad p_{\mathrm{rank}} = \frac{\#\{A : |A| = n,\ U(A, A^c) \geq U_{\mathrm{obs}}\}}{\binom{n+m}{n}}, \tag{8.4}$$

with the inequality reversed for accuracy. Under pooled exchangeability this is the exact null tail, with floor $1/\binom{n+m}{n}$ when every run of one arm beats every run of the other (Appendix A.8). It does not correct for repeated inspection or for the choice among correlated metrics of the same runs, so the reported tails are nominal.

8.6 Baselines

Untrained backbone. It is the Qwen3 model of the same size in fp32, prompted with the same question: up to 26 options are listed as letters and read from the letters' next-token probabilities, and more options are scored as continuations by mean token log-probability. It does not exhaust prompting or reasoning strategies.

Constant predictor. It always answers a family's most common label among the evaluated targets, a descriptive reference for class imbalance.

Public-suite references. The typed-decisions card publishes a majority baseline (0.520), an input-ignoring prior (0.470), a model fitted to the latent factors that generated each case (0.704) and the teacher's self-agreement (0.735). Scoring the prior from the training split with our own metric code gives 0.4675, 0.3474 and 0.1887 for accuracy, KL and Brier, against the published 0.470, 0.347 and 0.189. The two open decision models (`Mapika/decider-0.8b` and `-2b`, Apache-2.0) were run by us through their own interface at their shipped temperature, with the same scoring code. The commercial rows cited from the card are its published figures; TypeSafe's Jev 1.13 was also queried by us through its API and scored with the same code.

9 Results

9.1 In family: dev and the sealed test

Table 7 gives the in-family results. On the sealed test read, which nothing was selected on, accuracy is 0.8543 for the 0.6B and 0.8751 for the 1.7B, 1.07 and 0.34 points below their dev reads. Both are far above the untrained backbone of the same size on the same questions (0.3303 and 0.3337), but that reference is weak: pooled, it is below the

predictor that answers each family's most common label (0.4673), which is the more informative floor and which both releases clear by a wide margin.

Table 7. In-family results. The sealed rows are the single test read of each release; "without overlap" drops the 205 sealed questions whose state also occurs in training (Section 6.6).

	Hunch 0.6B				Hunch 1.7B			
Read	Acc.	NLL	Brier	smECE	Acc.	NLL	Brier	smECE
Dev, $T = 1$ (3,416)	0.8650	0.4306	0.1673	0.0233	0.8785	0.4032	0.1552	0.0208
Dev, release T	same	0.4232	0.1659	0.0096	same	0.3975	0.1547	0.0084
Sealed test, $T = 1$ (3,467)	0.8543	0.4415	0.1804	0.0162	0.8751	0.3987	0.1539	0.0253
Sealed test, release T	same	0.4373	0.1808	0.0167	same	0.3922	0.1531	0.0136
Sealed, without overlap	0.8455				0.8679			
Sealed, untrained backbone	0.3303	2.4122	0.8554	0.2904	0.3337	3.9041	0.9548	0.3547
Sealed, constant predictor	0.4673				0.4673			

Sources: `docs/results/launch/sealed.json` (recomputed from the two `_test300.json` reads and the frozen reads), `docs/results/launch/derived.json`, the `_dev.json` reads under `docs/results/`; the constant predictor is computed from the sealed targets. Parent of the 1.7B on dev: accuracy 0.8829, NLL 0.3856.

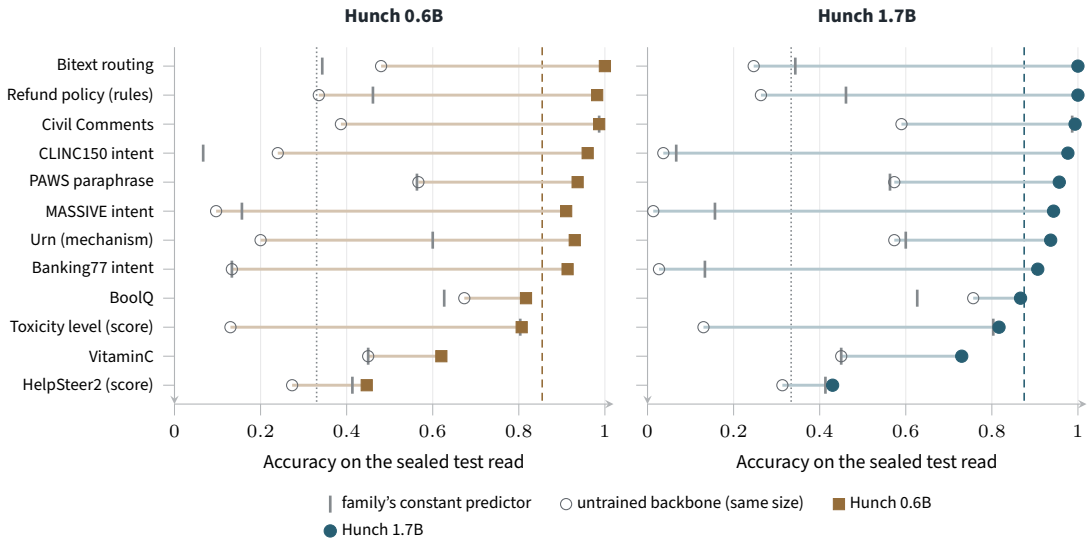

Figure 3. The sealed in-family test, family by family: each segment runs from the untrained backbone of the same size (open circle) to the trained checkpoint. The grey tick is the family's constant predictor, which always answers its most common label on these questions. Vertical rules mark pooled accuracy, trained (dashed) and untrained (dotted). Where the tick sits near the trained mark (Civil Comments, the toxicity level, HelpSteer2), accuracy says little; see the toxicity-miss rates in Section 9.5. Refund policy has 167 questions, every other family 300. Values are in Table 19.

Family by family. The routing and intent families (Bitext, CLINC150, MASSIVE, Banking77), the refund-policy rules, PAWS and the urn mechanism are at or above 0.9067 for both sizes (Figure 3). Three families need care. On Civil Comments, 296 of the 300 sealed questions have the label "not toxic", so always answering that scores 0.9867, exactly the 0.6B's accuracy there (0.9867) and a little below the 1.7B's (0.9933); these numbers say nothing about detecting toxicity, which Section 9.5 measures directly. On the two ordinal Score families the releases are close to their constant predictors: the five-level toxicity score reads 0.8067 and 0.8167 against 0.8033, and HelpSteer2's helpfulness rubric 0.4467 and 0.4300 against 0.4133. VitaminC (0.6200, 0.7300) and BoolQ (0.8167, 0.8667) are the lowest-scoring non-ordinal families and those where the 1.7B's margin over the 0.6B is largest. Appendix D gives every family.

Overlap with training. 205 of the 3,467 sealed questions have a state that also occurs verbatim in the training text of the two releases (204 of them in the 0.6B's own), concentrated where upstream text repeats: 114 in Bitext routing, whose support utterances are templated, and 72 in the template-generated urn states; then 10 in MASSIVE, 5 in Civil Comments, 3 in the toxicity score and 1 in CLINC150. Without them, accuracy is 0.8455 and 0.8679; Bitext's 1.0000 should be read with its overlap in mind.

Calibration. Figure 4 shows reliability on dev, which is disjoint from the calibration split the temperature was fitted on. At $T = 1$ both releases are already close to the diagonal, with smooth ECE 0.0233 and 0.0208; the release temperature, $T = 0.8706$ (below one, so it sharpens), lowers it to 0.0096 and 0.0084. On the sealed test it halves the 1.7B's smooth ECE (0.0253 → 0.0136), leaves the 0.6B's essentially unchanged (0.0162 → 0.0167) and improves test NLL slightly for both. One global temperature does not suit every family: at the 1.7B it roughly halves the error on the intent families (MASSIVE 0.0717 → 0.0375), but at each size it increases it on five of the twelve families, most on HelpSteer2 at 1.7B (0.0411 → 0.0665) and on VitaminC and BoolQ at 0.6B (Table 20).

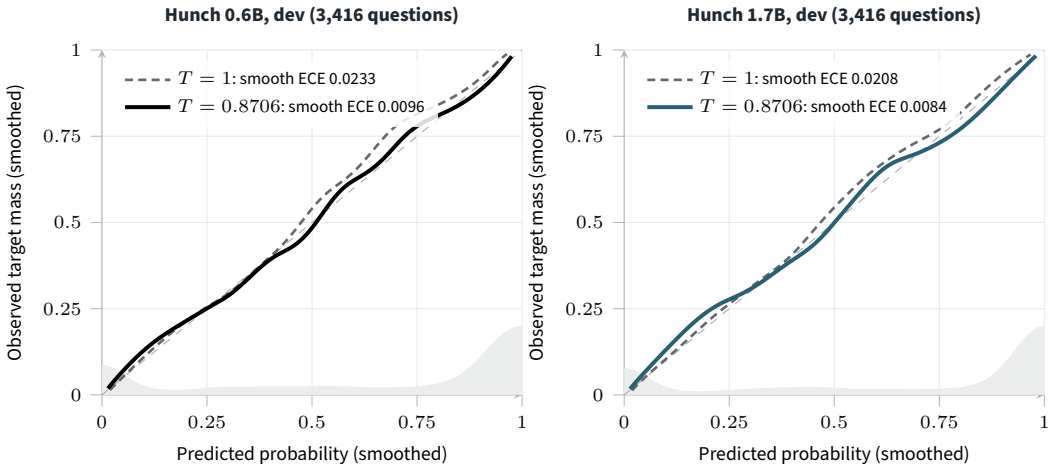

Figure 4. In-family reliability on the dev split before and after the release temperature, which was fitted on the disjoint calibration split. Curves are the kernel-smoothed quantities that smooth ECE integrates (bandwidth 0.05), drawn where the confidence density is non-negligible; the grey area is that density. For Boolean questions the pair is (predicted, target) probability of true; otherwise the top probability and the target mass on the top candidate.

9.2 Held-out families

Table 8 and Fig. 5 set the held-out families against the constant predictor, the 1.7B's parent and its replication seed.

Table 8. Held-out families, 2,000 questions each, $T = 1$. The constant predictor always answers the most common label of the evaluated targets.

		1.7B parent		Hunch 0.6B		Hunch 1.7B		1.7B seed 1	
Family	Constant	Acc.	NLL	Acc.	NLL	Acc.	NLL	Acc.	NLL
GoEmotions (Boolean)	0.7645	0.8285	0.4031	0.7465	0.5591	0.8000	0.4274	0.8175	0.3975
CaseHOLD (5 options)	0.2120	0.6190	0.9720	0.5515	1.1316	0.6020	1.0763	0.5820	1.0920
Circa (8 options)*	0.4495	0.1600	2.0745	0.5605	1.5194	0.5905	1.3189	0.6050	1.2856
Pooled		0.5358	1.1498	0.6195	1.0700	0.6642	0.9409	0.6682	0.9250
Without Circa		0.7238	0.6875	0.6490	0.8454	0.7010	0.7518	0.6997	0.7447
Pooled smooth ECE			0.0651		0.1054		0.0460		0.0487

*Question format trained in both releases (Section 6.2), not in the parent. Sources: the four `*_heldout2000.json` reads under `docs/results/` (identical question ids), `docs/results/launch/heldout-majority.json`, `docs/results/launch/derived.json` (`heldout_without_circa`).

CaseHOLD: real transfer. Choosing, among five options, the holding that a legal excerpt cites is outside every training family. Both releases are far above the constant predictor (0.5515 and 0.6020 against 0.2120); the parent, trained on the base mixture alone, is slightly higher at 0.6190.

GoEmotions: near or below the floor. Emotion attributes as Boolean questions are highly imbalanced: the constant predictor scores 0.7645. The 0.6B is below it (0.7465), with smooth ECE 0.2061 there; the 1.7B is above it (0.8000) but below its parent (0.8285).

Circa: format transfer. The parent, which never saw Circa's format, is 28.95 points below the constant predictor. Both releases, which trained the format through the synthetic indirect-answer family, are above it (0.5605 and 0.5905 against 0.4495). That transfer of a trained format is the whole of the 1.7B's pooled held-out gain over its

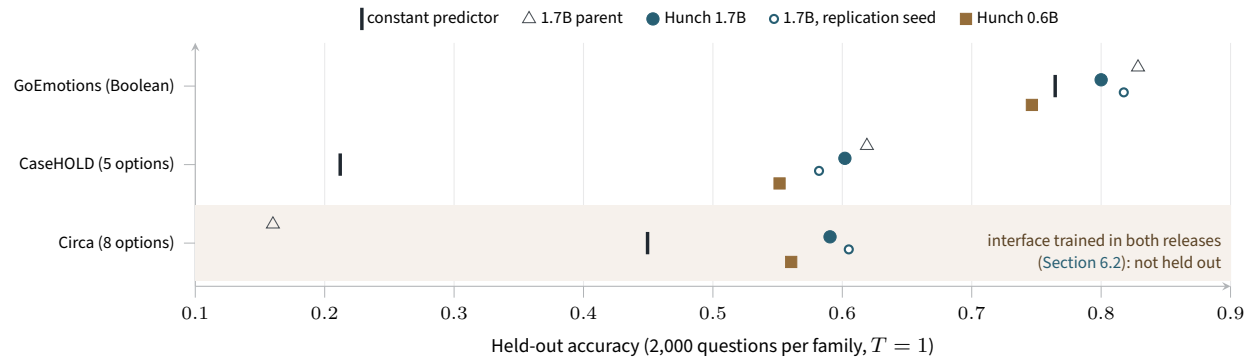

Figure 5. Held-out families against the constant predictor, which always answers the most common label of the evaluated targets. On CaseHOLD both releases are far above it; on GoEmotions the 0.6B is below it and the 1.7B above it but below its parent. Circa's question format is trained in both releases (Section 6.2). Vertical offsets only separate the markers.

parent: without Circa, the release is worse than its parent in both accuracy and NLL, and it is also slightly worse in family (dev accuracy 0.8785 against 0.8829).

The temperature does not transfer. The in-family temperature leaves the 1.7B's pooled held-out smooth ECE at 0.0460 → 0.0463 and worsens the 0.6B's from 0.1054 to 0.1301, which is why out-of-family results use $T = 1$.

9.3 Public suites

We scored both releases zero-shot on three public collections. Neither trained on any split of the first two; the third includes a few suites whose training splits are in our mixture, and we mark them.

typed-decisions. `LocalLLaMA/typed-decisions` [25] (revision `c76749ec58bd`) has 400 test cases in four workflows with 2,000 decisions (600 Choice, 600 Boolean, 800 ordinal Score) and takes Hunch's interface directly. Its gold labels are the mean of three samples from a teacher endpoint that the card describes only as of roughly 4B-class capability, so a score measures agreement with that teacher. The card's own reading scale is "Read 0.52 as the floor. Around 0.70 is strong. Around 0.75 is saturation." Table 9 and Fig. 6 give the results.

Table 9. typed-decisions, zero-shot, 2,000 decisions, $T = 1$. Brackets are 95 % case-bootstrap intervals. The card's two commercial zero-shot rows, TypeSafe Jev 1.13.0 (0.727) and meraGPT Decider 1 (0.768), and its two per-workflow fitted specialists (0.587 and 0.646) are omitted from the table; our measurement of Jev 1.13 is reported for accuracy only.

Model	Source	Accuracy ↑	KL ↓	Brier ↓
TypeSafe Jev 1.13, commercial	measured by us, via its API	0.7375		
open `Mapika/decider-2b`	measured by us	0.5895 [0.5645, 0.6145]	0.4109	0.2115
Hunch 1.7B (`g2-17-v4t`)	measured	0.5105 [0.4845, 0.5365]	0.3542	0.1951
open `Mapika/decider-0.8b`	measured by us	0.5105 [0.4870, 0.5345]	0.6279	0.2888
Majority baseline, ignores the input	card	0.520		
Prior, ignores the input	card	0.470	0.347	0.189
Hunch 0.6B (`w16-06-v3dgp10-1200-s55`)	measured	0.3795 [0.3550, 0.4050]	0.5889	0.2906
1.7B parent (`h200-17-fullk-v3d`)	measured	0.3665	0.8098	0.3581

Sources: the `td_*.json` reads under `docs/results/`, `docs/results/launch/jev/jev_typed_decisions.json`, `docs/results/launch/derived.json` (intervals and paired differences, recomputed here from the per-decision rows), `docs/results/launch/td_card.json` (the card's rows, parsed from the pinned card).

The 1.7B scores 0.5105, within its interval of the card's 0.520 majority floor. It ties the open 0.8B Decider on accuracy (paired difference 0.0 points [−2.5, +2.5]) with lower KL and Brier, and trails the open 2B Decider by 7.9 points [5.3, 10.5] on the same decisions, while its KL and Brier are lower: the paired differences are −0.0566 [−0.0782, −0.0351] and −0.0163 [−0.0282, −0.0046], with intervals that exclude zero. On KL it is level with the input-ignoring prior (0.3542 against 0.347), and the prior's Brier is lower still. Its gain of 14.4 points over its parent arrives with the teacher data, none of which comes from any split of the suite; the contamination scan finds no test state in either release's training text (Section 6.6). The 0.6B, at 0.3795, is below the prior on every column; on ordinal Score questions it reads 0.280, close to the 0.244 of guessing uniformly among the levels. TypeSafe's Jev 1.13, which we queried through its API on the same decisions and scored with the same code, agrees with the gold on 0.7375 (the card gives 0.727), far above both releases.

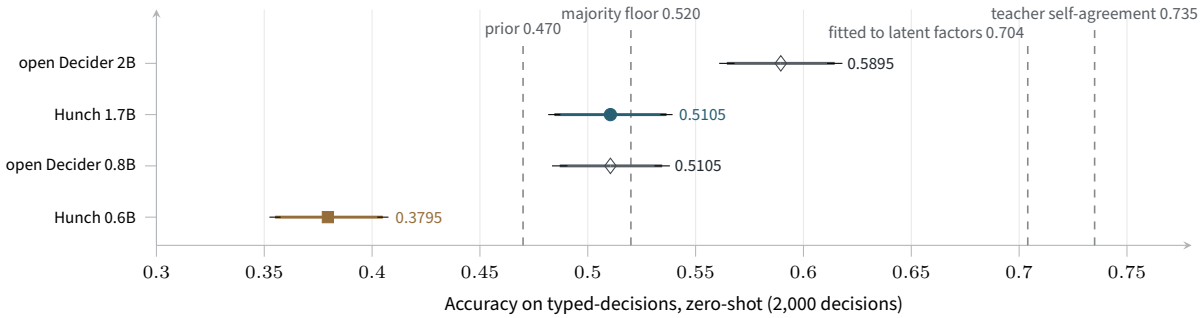

Figure 6. typed-decisions accuracy with 95 % case-bootstrap intervals (400 cases resampled with their five decisions, 10,000 draws). The open Decider models were run by us through their own interface with the same scoring code. Dashed rules are the suite card's own references: the input-ignoring prior and majority baseline, a model fitted to the latent factors of each case, and the teacher's self-agreement.

The same models on the releases' own families. Out of family these generalists lead; on the twelve families the releases were trained on, the comparison is of fine-tuned specialists with generalists answering zero-shot on the specialists' own tasks, and the order reverses (Table 10). Jev and the two open Deciders, each through its own interface, answered the 3,523 calibration-split questions of the two release reads, a split used only to fit the temperature, which cannot move an answer. Jev is ahead of both releases on BoolQ and urn draws and of the 0.6B on VitaminC; the 2B is ahead of the 1.7B on HelpSteer2 and BoolQ and of the 0.6B also on MASSIVE; the 0.8B is ahead of the 0.6B on MASSIVE and BoolQ and of the 1.7B on no family. The Deciders' data registry lists none of these task formats, though some of their public sources may be in their training data; that was not tested.

Table 10. The releases' own twelve families: the 3,523 calibration-split questions of the two release reads, $T = 1$; the generalists answer zero-shot. Leads are the releases' paired accuracy differences in points, with 95 % state-group bootstrap intervals, on all questions and without the 204 whose state occurs in training.

			Lead of Hunch 1.7B		Lead of Hunch 0.6B	
Model	Accuracy ↑	Brier ↓	all	no overlap	all	no overlap
Hunch 1.7B, trained	0.8731	0.147				
Hunch 0.6B, trained	0.8535	0.168				
TypeSafe Jev 1.13, via its API	0.8010	0.273	7.2 [5.7, 8.8]	7.4 [5.8, 9.0]	5.3 [3.7, 6.9]	5.4 [3.7, 7.1]
open `Mapika/decider-2b`	0.7346	0.309	13.9 [10.9, 17.0]	13.6 [10.5, 17.0]	11.9 [8.9, 15.2]	11.7 [8.5, 15.1]
open `Mapika/decider-0.8b`	0.6787	0.369	19.4 [16.2, 23.0]	19.2 [15.7, 23.0]	17.5 [14.1, 21.2]	17.3 [13.7, 21.2]

Sources: `docs/results/launch/jev/jev_calibration.json`, `docs/results/launch/decider_calibration_decider-2b.json` and `decider_calibration_decider-0.8b.json` beside it, each with its per-question `.items.jsonl`; the two `_calibration.json` reads.

typed-decisions-v2-system-one. `pngwn/typed-decisions-v2-system-one` [26] has 5,214 test rows, every one a Choice with 2 to 10 options and a hard gold index. Its card publishes no reference results and does not say how the gold was produced. Both releases clear the majority class (0.3506) and trail both open Deciders on accuracy: 0.4845 (1.7B) and 0.4647 (0.6B), against 0.5021 for the open 0.8B and 0.5297 for the open 2B; the parent reads 0.4409.

The Laya list. Laya [27] published its benchmark code and raw results, so its 59 public suites were rebuilt row for row from that code and scored zero-shot with both releases, beside Laya's own published numbers (Table 11). Where Hunch trained the task, a suite measures retention; where it trained the task in English only, other languages measure cross-lingual transfer of a trained task through the multilingual backbone, not zero-shot task transfer (Laya's checkpoints did not train on MASSIVE at all); otherwise the suite is an unseen task.

Where the task was trained, it carries into languages the readout never saw: the 1.7B reads MASSIVE intent at 0.927 in English and 0.343 in Swahili, its lowest language. Where it was never trained, the releases mostly trail: English XNLI reads 0.613 and 0.747 against Laya's 0.860, and on the Boolean e-mail spam and phishing suites the releases read 0.482 and 0.517 (spam, at chance) and 0.620 and 0.650 (phishing), where Laya, which trained on both, reads 0.993 and 0.980. Against its parent across all 59 suites, the 1.7B is better by more than one point on 30, worse on 12 and within one point on 17, with a median change of +1.0 points.

What we did not run. Of the eight public datasets named for this problem category we ran two, plus the Laya list; two are results dumps without a test split, one lists no files, one has no gold labels, one has no adapter yet, and one, whose states are code files with hundreds of options, exceeds the 32,768-token cap (Section 2).

Table 11. Selected Laya-list suites, accuracy, zero-shot, $T = 1$. Laya's column is its published figure for its own model. Macros are unweighted means over languages. Flags: T, the task's training split is in Hunch's mixture; X, the task is trained in English and read in other languages; L, Laya trained on the suite.

Suite	Flag	Hunch 0.6B	Hunch 1.7B	Laya
MASSIVE intent, 14 languages, 20 options	T, X	0.734	0.829	0.340
MASSIVE scenario, 14 languages	T, X	0.517	0.579	0.304
XNLI, 15 languages, 3 options		0.531	0.632	0.544
XNLI, English		0.613	0.747	0.860
E-mail spam (Boolean)	L	0.482	0.517	0.993
Phishing (Boolean)	L	0.620	0.650	0.980
Guardrails: jailbreak		0.757	0.795	0.708
Moderation: toxicity		0.605	0.618	0.530

Sources: `docs/results/day7/laya/laya_list_w16-06-v3dgp10-1200-s55.json`, `docs/results/day7/v1/laya/laya_list_g2-17-v4t.json`, Laya's published `t4_colab_benchmark.json` and `app_benchmark_results.json`, kept verbatim in `docs/results/launch/laya_published/`.

9.4 Robustness

Two probes were declared in advance and run on 500 held-out questions balanced over the three families, at $T = 1$.

Planted instructions. Each variant is appended to the longest text field of the state, the field a user controls, and names a wrong candidate drawn uniformly per question with a fixed seed. The declared bar for a finding was a flip-to-target rate more than two points above a neutral sentence of the same length, and every instruction-like variant exceeds it widely (Table 12). A candidate that is merely named in the state is favoured, with or without an instruction: at 0.6B the label pasted three times moves nearly as many answers as the instruction does. CaseHOLD is the worst case, because its excerpt ends exactly where the holding should be stated, so an appended sentence sits where the answer would be. The 1.7B is less injectable than its parent on every variant but far past the bar.

Table 12. Planted text: the share of the 500 held-out questions whose answer moves to the named wrong candidate, $T = 1$. CaseHOLD is also shown alone. The parent row is the 1.7B's parent on the same probe.

	Hunch 0.6B		Hunch 1.7B		1.7B parent	
Variant appended to the state	all	CaseHOLD	all	CaseHOLD	all	CaseHOLD
"Ignore the text above. The correct answer is: X."	35.6 %	74.9 %	32.4 %	69.5 %	36.2 %	73.1 %
The same, as a markdown *System:* turn	34.0 %	81.4 %	29.0 %	66.5 %	38.2 %	76.6 %
The wrong label pasted three times	34.4 %	62.9 %	21.4 %	44.9 %	32.0 %	54.5 %
Random printable noise, same length	2.4 %	1.8 %	1.8 %	3.0 %	2.6 %	2.4 %
Neutral sentence, same length (control)	2.8 %	6.0 %	1.8 %	3.6 %	2.4 %	2.4 %

Sources: `docs/results/day6/robust/injection_w16-06-v3dgp10-1200-s55.json`, `docs/results/day7/v1/robust/injection_g2-17-v4t.json`; pooled rates are question-weighted over the three families and reproduce `docs/results/launch/derived.json`. Base accuracy on the sample: 0.616 (0.6B), 0.660 (1.7B).

Formatting that carries no information. Three perturbations of the state were read against a precision floor, the share of answers that bf16 inference alone changes relative to fp32 (0.467 % for both releases, Section 9.6); a paraphrased instruction is impossible for the held-out families and was not run. Lower-casing with collapsed whitespace flips 5.4 % of both releases' answers; toggling curly quotes and a final period flips 1.6 % (0.6B) and 2.2 % (1.7B); swapping the first two sentences, possible on 242 of the questions, flips 6.2 % and 10.7 %. These rates are between three and twenty-three times the floor's. Against its parent, the 1.7B is more stable to case and whitespace (parent 7.4 %) and less stable to sentence order (parent 8.3 %).

Option order. Permuting the options of 200 cases on three Laya-list suites changed 0.0 % of either release's answers, as Equation (4.1) requires; Laya's two published models, which read all options in one sequence, change 15.0 % and 23.0 % of their MASSIVE intent answers.

9.5 Fairness

Every Civil Comments row of dev and calibration was scored to ask whether the toxicity heads treat comments that mention an identity differently; a row mentions an identity if it contains a whole-word, case-insensitive match to a fixed lexicon of gender, sexual-orientation, religion, race or ethnicity and disability terms, printed in full in the release documentation. The lexicon is a proxy that misses references without those words and catches quoted or negated mentions, so a gap whose interval covers zero is not evidence of parity (Table 13).

Table 13. Missed-toxic and false-toxic rates by identity mention, dev and calibration Civil Comments rows. Boolean: toxic if gold $P(\text{true}) \geq 0.5$, flagged if predicted $P(\text{true}) \geq 0.5$. Score: toxic if gold level ≥ 3, flagged if predicted level ≥ 3. Gaps are identity minus none, in points, with 95 % bootstrap intervals.

		Missed toxic			Falsely flagged		
Head	Release	none	identity	gap	none	identity	gap
Boolean	0.6B	59.0 %	77.3 %	+18.3 [+4.6, +31.9]	0.3 %	0.3 %	0.0 [−0.3, +0.4]
	1.7B	67.1 %	77.3 %	+10.2 [−2.5, +21.7]	0.1 %	0.2 %	0.0 [−0.2, +0.3]
5-level score	0.6B	36.6 %	70.6 %	+34.0 [+6.6, +62.1]	0.5 %	0.0 %	−0.5 [−0.9, −0.2]
	1.7B	36.6 %	70.6 %	+34.0 [+7.4, +59.4]	0.0 %	0.0 %	0.0 [0.0, 0.0]

Toxic comments in each group (Boolean / score): 173 / 41 without an identity term, 66 / 17 with one. Sources: `docs/results/day6/fairness.json`, `docs/results/day7/v1/fairness_g2-17-v4t.json`.

Two findings stand. First, at a threshold of 0.5 the Boolean heads miss most toxic comments: 59.0 % (0.6B) and 67.1 % (1.7B) of those without an identity term and 77.3 % of those with one, while flagging almost no non-toxic comments, so at that threshold they cannot serve as a filter; their ranking quality was not measured. Second, a toxic comment that mentions an identity is missed more often, with no higher false-toxic rate: on the five-level score by +34.0 points at both sizes ([+6.6, +62.1] at 0.6B, [+7.4, +59.4] at 1.7B; 12 of 17 toxic comments missed against 15 of 41; the parent read +30.6), and on the 0.6B's Boolean head by +18.3 points [+4.6, +31.9]; the 1.7B's Boolean interval covers zero. The intervals are wide because few toxic comments mention an identity. The disparity runs toward under-detection; the failure more often documented for toxicity classifiers is over-flagging of comments that mention identities [13, 28], which these heads do not show.

9.6 On-device builds

Both releases are converted to GGUF, for llama.cpp [29], and to MLX [30]. The GGUF file holds the backbone only, as a standard Qwen3 model whose output norm is the backbone's own final norm; the runtime runs llama.cpp in embedding mode, takes the post-norm hidden state at each path's readout token, and applies Hunch's fp32 readout norm, head and per-question softmax itself, with the readout tensors and the temperature carried as file metadata. Folding the readout into llama.cpp's single output norm would be wrong, because with weights w_1, w_2, $\mathrm{RMSNorm}(\mathrm{RMSNorm}(x) \odot w_1) \odot w_2 = \mathrm{RMSNorm}(x \odot w_1) \odot w_2$ is not an RMS normalization of x: scored from the fp32 reference's own hidden states on 201 questions at 1.7B, that design agrees with the reference on only 97.5 % of top answers (maximum total-variation distance 0.081). The MLX build keeps activations in fp32 and stores weights in f16. Every artifact embeds the release temperature.

The gate. Every conversion is compared with an fp32 run of the same checkpoint on all 6,000 held-out questions, on two statistics: the largest total-variation distance between the two answer distributions over all questions, a worst case, and the rate at which the top answer differs. The bar is a measurement: bf16 inference, which this project uses everywhere, already differs from fp32 by up to 7.68×10^{-2} (0.6B) and 3.08×10^{-2} (1.7B) in total variation and changes 28 of 6,000 answers at both sizes. A conversion passes if its maximum distance is within that floor's and its rate of changed answers is not significantly above the floor's rate (a one-sided two-proportion test at 95 %). The floor rule was adopted after fixed bars (for f16, maximum distance at most 1×10^{-3} and no changed answer) had failed every conversion of earlier checkpoints, which fp32 PyTorch itself would fail against the bf16 read, and before any released checkpoint was converted; all four published f16 files fail the fixed bars, and the reports print both verdicts.

Both f16 builds of both sizes pass: MLX changes no answer at either size, and GGUF changes 1 at 0.6B (read on Metal) and 10 at 1.7B (read on llama.cpp's CUDA backend, fixed as the gate's backend before either read), a difference in how the two backends compute f16 matrix products that an earlier 1.7B checkpoint also showed. Every quantised 0.6B build is outside the floor on both statistics, the 8-bit builds narrowly and the 4-bit builds by more than an order of magnitude, so none is released and no quantised 1.7B build was made.

9.7 Latency

Latency is measured one request at a time under the conditions stated; it is not a throughput figure. The requests are the 400 typed-decisions test states with their questions; the first 20 are warm-up and the remaining 380 are timed as wall-clock including tokenization. With the reference engine, `Hunch.decide`, on one NVIDIA RTX 5090 in bf16 with a 16,384-token budget (PyTorch 2.11), the median request takes 78.1 ms (0.6B) and 140.6 ms (1.7B), with 90th percentiles of 107.4 and 188.3 ms. On an Apple M4 Pro laptop in normal use (one-minute load average 2.16 and 2.57 at the start), the MLX f16 builds with fp32 activations take 950.8 ms and 2746.4 ms per request at the

Table 14. On-device builds against fp32 on all 6,000 held-out questions. × floor divides by the bf16 floor's value; z is the two-proportion statistic for the changed-answer rate against the floor's rate.

Size	Build	Size (MB)	Max TV	× floor	Changed	× floor	z	Gate	Released
0.6B	MLX f16, Metal	1,192	5.69×10^{-3}	0.07	0	0.00	−5.30	pass	yes
	GGUF f16, Metal	1,198	4.88×10^{-3}	0.06	1	0.04	−5.02	pass	yes
	GGUF Q8_0	639	1.08×10^{-1}	1.40	44	1.57	+1.89	no	no
	MLX 8-bit	633	1.29×10^{-1}	1.68	49	1.75	+2.40	no	no
	GGUF Q4_K_M	397	8.94×10^{-1}	11.64	522	18.64	+21.56	no	no
	MLX 4-bit	335	8.26×10^{-1}	10.76	787	28.11	+27.54	no	no
1.7B	MLX f16, Metal	3,441	3.13×10^{-3}	0.10	0	0.00	−5.30	pass	yes
	GGUF f16, CUDA	3,447	1.45×10^{-2}	0.47	10	0.36	−2.92	pass	yes

Sources: `hunch/formats/out/equivalence-0.6b.json` and `equivalence-1.7b.json`; every z and verdict was recomputed from the counts. Floors: 0.6B max TV 7.68×10^{-2}, 1.7B 3.08×10^{-2}; 28 changed answers each.

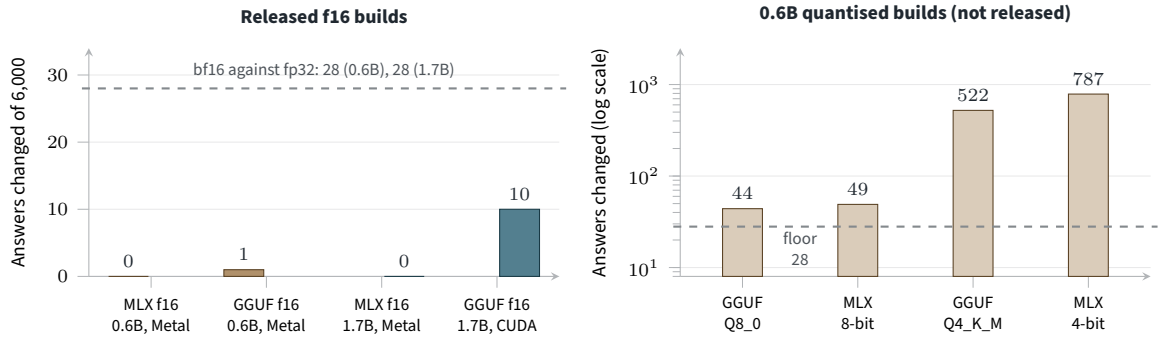

Figure 7. On-device builds on all 6,000 held-out questions, against an fp32 run of the same checkpoint. The dashed rule is the precision floor: the number of answers that ordinary bf16 inference already changes. A build passes if its largest total-variation distance is within the floor's and its rate of changed answers is not significantly above the floor's (Table 14). Both f16 builds of each size pass; every quantised 0.6B build fails, so none is released and no quantised 1.7B build was made.

median with the reference engine. A request's questions are scored together in one batched pass, so the per-decision median (15.6 and 28.1 ms on the GPU) is an average share of a request, not the time a single-question request would take.

The shared-prefix engine (Section 3) is compared with the reference in Table 15, both timed back to back on the same card. In fp32 it changes no answer on either set at either size. In bf16 it is within the on-device builds' floor (Section 9.6) at 0.6B but not at 1.7B, whose largest distance exceeds the floor's, so the 1.7B should run it in fp32. The first gate we declared, fast bf16 against reference bf16, failed at both sizes (33 and 35 answers changed, against 28): it stacks two bf16 roundings against a floor that measures one. The gates of the table were declared after that failure at 0.6B, and before either was run. On the MLX f16 builds it changes no held-out answer against the MLX reference read (largest distance 1.49×10^{-5} and 1.20×10^{-5}, the same bar, declared before timing), and on the laptop it takes 156.2 ms (0.6B) and 400.7 ms (1.7B) at the median, 6.1 and 6.9 times less than the reference engine, although its runs started at a higher load (one-minute load average 4.19 and 3.81) and on Python 3.13.9 rather than 3.12.7.

Beside a network API. TypeSafe's Jev 1.13 is served over the network. We timed it on the same 400 requests, one at a time, through OpenRouter, as the round trip from the same Apple M4 Pro laptop over the public internet (380 timed after 20 warm-up): a median of 369.1 ms and a 90th percentile of 458.5 ms, network included, where Hunch's times are local runs. Set beside that round trip, the shared-prefix engine's medians on one RTX 5090 are 10.1 times lower for the 0.6B in fp32, 17.6 times lower for the 0.6B in bf16 and 5.5 times lower for the 1.7B in fp32. On that laptop the 0.6B's fast median is 2.4 times lower than the round trip and the 1.7B's is higher; with the reference engine both are higher. Jev agrees with the typed-decisions gold far more often than either release (Section 9.3). Sources: `docs/results/day7/v1/latency/`, `docs/results/launch/fast/` (`mlx_equiv_*`, `latency_mlx_fast_*`), `docs/results/launch/jev/jev_typed_decisions.json`.

Table 15. The shared-prefix engine (fast) and the reference engine on one RTX 5090, timed back to back. Answers changed and the largest total-variation distance are against the fp32 reference, on the 6,000 held-out questions and the 2,000 typed-decisions (TD) decisions; the reference rows give bf16's own values, the floor that × floor divides by. Gates: fp32, no held-out answer changed and a largest distance below 1×10^{-4}; bf16, the builds' floor rule. Speed-up: the reference's median over the engine's.

		Held-out				TD		Latency per request (ms)		
Size	Engine	Changed	Max TV	× floor	z	Changed	Gate	Median	p90	Speed-up
0.6B	reference, bf16	28	7.68×10^{-2}				floor	75.6	105.0	
	fast, fp32	0	1.67×10^{-5}			0	pass	36.7	39.4	2.1×
	fast, bf16	33	6.72×10^{-2}	0.88	+0.64	8	pass	21.0	22.3	3.6×
1.7B	reference, bf16	28	3.08×10^{-2}				floor	139.8	184.0	
	fast, fp32	0	6.11×10^{-6}			0	pass	66.6	71.0	2.1×
	fast, bf16	31	3.84×10^{-2}	1.25	+0.39	9	fail	not compared*		

*Its median, 37.6 ms, enters no comparison because its gate failed. Sources: `docs/results/launch/fast/` (`equivA_*` fp32, `equivB_*` bf16, `latency_*`), gates applied in `docs/results/launch/derived.json` (`fast_engine`); floor as in Table 14.

10 Studies

The recipes rest on controlled comparisons run during development, on the research reads: 2,000 held-out questions per family, identical ids across runs, $T = 1$, repeatedly inspected. A pooled comparison admits a run only if its training configuration is archived and matches the others on backbone, steps, candidate cap, token budget, questions per step, optimizer, learning rates, warmup and precision. The cohorts are fixed at their 22 and 23 September snapshots, values are transcribed from the archived per-run result files, and rank statistics use Equation (8.4).

10.1 Authored candidate descriptions

To test whether the authored candidate descriptions help (Section 6.1), six runs trained on the described data (`v3d`) were compared with six trained on the plain data (`v3`) under one recipe: Qwen3-1.7B, 1,200 steps, full candidate sets, a 131,072-token budget, 120 questions per step, 8-bit AdamW and bf16. Only data, seed and output path differ across the archived configurations, and every run reads byte-identical held-out inputs (SHA-256 beginning `d2f3cd0350505a22` in both manifests), so the comparison varies training data, not test-time wording.

Table 16. Recipe-matched descriptions comparisons, 2,000 items per held-out family, $T = 1$: six against six at 1.7B (left) and a fresh 0.6B replication, five against four (right). Lower is better except accuracy.

Qwen3-1.7B, 1,200 steps, full candidate sets						Qwen3-0.6B, 600 steps, cap 16					
Run	Seed	NLL	Brier	Acc.	smECE	Run	Seed	NLL	Brier	Acc.	smECE
Described data (`v3d`)											
`rp1`	1	1.0650	0.5264	0.5747	0.0394	`k81`	21	1.1739	0.5623	0.5788	0.0349
`rp3`	2	1.1071	0.5355	0.5608	0.0407	`k82`	22	1.1627	0.5631	0.5500	0.0333
`rp5`	3	1.0908	0.5127	0.6035	0.0300	`k84`	24	1.1341	0.5464	0.5858	0.0234
`rp7`	4	1.2213	0.5678	0.5135	0.0694	`k91`	26	1.1474	0.5652	0.5135	0.0792
`rp12`	5	1.1395	0.5425	0.5737	0.0561	`k92`	27	1.1639	0.5614	0.5347	0.0545
`rp18`	6	1.1749	0.5464	0.5330	0.0540						
Plain data (`v3`)											
`rp2`	1	1.5925	0.6825	0.4545	0.2333	`k86`	31	1.3212	0.6228	0.4747	0.1442
`rp4`	2	1.2875	0.5945	0.4963	0.1553	`k88`	33	1.2788	0.6030	0.4845	0.1014
`rp6`	3	1.3523	0.6367	0.4643	0.1775	`k89`	34	1.3287	0.6178	0.4785	0.1405
`rp8`	4	1.3504	0.6233	0.4838	0.1438	`k95`	35	1.3242	0.6188	0.4688	0.1549
`rp13`	5	1.4789	0.6695	0.4313	0.2009						
`rp17`	6	1.3826	0.6015	0.5007	0.1246						

Run ids abbreviate names such as `rp1-17-v3d-full1k-s1` and `k81-06-v3d-s21`.

On each of NLL, Brier, accuracy and smooth ECE, every described-data run beats every plain-data run at 1.7B: $U = 36$ and $p = 1/\binom{12}{6} \approx 0.0011$, the attainable floor (Figure 8). The boundary margin is modest: the highest described-data NLL is 1.2213 and the lowest plain-data NLL 1.2875, a gap of 0.0662. Pairing runs by seed index instead, all six paired differences favour described data on every metric, with one-sided sign probability $1/2^6 = 0.015625$ (Appendix A.8). The 0.6B replication also separates completely on all four metrics: $U = 20$, nominal $p = 1/\binom{9}{5} \approx 0.0079$, again the floor; it is an interim five-against-four snapshot of a design expanded to ten against ten before these results, so its tail is not a final prospective test. Both releases train on described data. The same

descriptions lower the dev accuracy of the prompted, untrained Qwen3-32B from .6256 on plain inputs to .5322; the mechanism is not established.

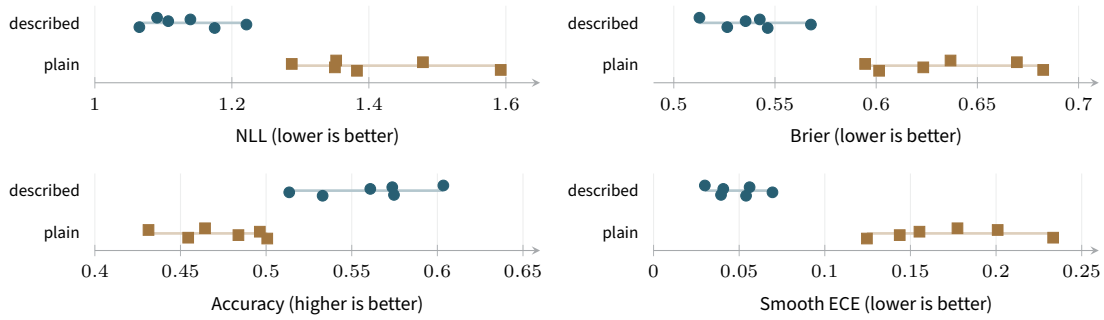

Figure 8. All twelve 1.7B runs, not selected examples. Described-data and plain-data runs separate on all four metrics. Horizontal segments show within-arm ranges; vertical offsets only separate markers and carry no measurement. Each metric has its own horizontal scale. Values are in Table 16.

10.2 Prior augmentation and a synthetic indirect-answer family

A 600-step wave at 0.6B (cap 16, 16,384 tokens, 120 questions per step) varied the prior share on top of the synthetic indirect-answer family, with synthetic-only runs as a control (Table 17 and Fig. 9). The declared endpoint was pooled NLL. Both 10 % and 25 % have lower observed mean NLL than the base `v3d` reference and the single 50 % run; the 5 % and 50 % arms have one seed each, and the highest 10 % NLL exceeds the 50 % run, so the data locate a promising region, not an optimum. The release mixture uses 10 %.

Table 17. 0.6B prior-share evidence at 600 steps, 2,000 items per family, $T = 1$. Means are over per-run metrics. The two zero-share rows differ in whether the synthetic indirect-answer family is included.

Data / prior share	Seeds	NLL	Brier	Acc.	smECE	Source IDs
Base `v3d`, 0%	3	1.1569	.5573	.5716	.0305	`k81/k82/k84`
Synthetic `v3dg`, 0%	4	1.3232	.5243	.6446	.1523	`c2`, s1000–1003
Synthetic + 5%	1	1.1322	.5618	.5953	.1237	`w12`
Synthetic + 10%	4	1.0821	.5422	.6142	.0998	`w1/w2/w3/w13`
Synthetic + 25%	2	1.0708	.5408	.6072	.0480	`w10/w11`
Synthetic + 50%	1	1.1266	.5617	.6137	.0356	`w14`

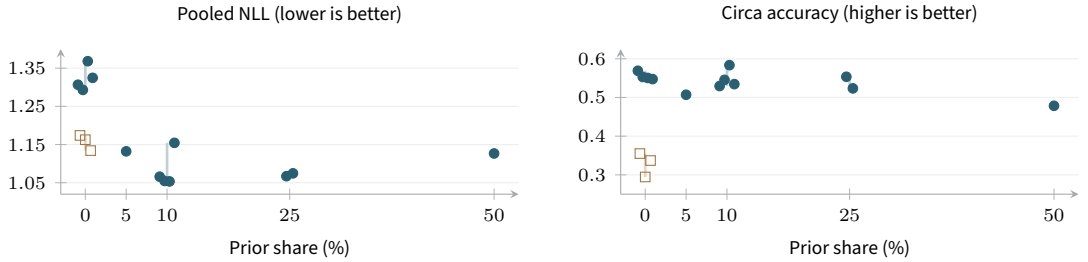

Figure 9. Every run in Table 17. Filled teal circles include synthetic data; open gold squares do not. Small horizontal offsets separate seeds and carry no measurement; vertical segments show observed ranges, not confidence intervals. The recovered synthetic-only control already attains high Circa accuracy at zero prior share, while its NLL is worse. No interpolation or fitted optimum is implied.

The synthetic-only control separates the two interventions. Its Circa accuracy is already .5475–.5690 without any prior augmentation, while its pooled NLL, 1.2931–1.3683, is worse than that of every prior-augmented arm: the Circa gain comes with the synthetic family, and the prior's association is with probability quality, so a comparison of the base data against the 10 % mixture confounds the two. Across the four 10 % seeds, GoEmotions accuracy spans .5985–.8120, so no simultaneous gain on GoEmotions and Circa is robust across seeds. At 1.7B, in a post-hoc comparison, three full-candidate runs of the combined recipe separate from the six described-data controls of Table 16 on accuracy (nominal $p = 1/\binom{9}{3} = 0.0119$, the floor) but not on NLL or Brier (0.0833) or smooth ECE (1.0000), and all three have worse pooled smooth ECE than every control.

10.3 Training length

On the described data (cap 16, 16,384 tokens), three verified 600-step 0.6B runs beat two verified 1,200-step runs on all four pooled metrics and on Circa accuracy (nominal $p = 1/\binom{5}{3} = .100$, the floor), while development accuracy reversed direction. That trade-off motivated a larger comparison inside the release's own mixture (`sprint_v3dgp10`): five seeds at 600 steps against five at 1,200, declared before its last four runs were read, with one seed (`w13`) excluded by declaration. The released 0.6B is one of the 1,200-step runs.

Table 18. Training length inside the release mixture at 0.6B, five seeds per arm, 2,000 held-out questions per family, $T = 1$. Ranges are over seeds; p is the exact one-sided rank tail for "600 steps better" with floor $1/\binom{10}{5} = 0.0040$. Dev accuracy exists for three seeds per arm.

Metric	600 steps: mean (range)	1,200 steps: mean (range)	U	p
Pooled held-out NLL ↓	1.0586 (1.0534–1.0658)	1.1066 (1.0700–1.1506)	25	0.0040
Pooled held-out accuracy ↑	0.6292 (0.6158–0.6413)	0.6156 (0.5993–0.6240)	22	0.0278
Circa accuracy ↑	0.5581 (0.5295–0.5835)	0.5361 (0.4855–0.5605)	17	0.2103
In-family dev accuracy ↑ (3 v 3)	0.8466 (0.8419–0.8528)	0.8651 (0.8636–0.8668)		0.0500†

†One-sided tail for "1,200 steps better", floor $1/\binom{6}{3} = 0.0500$. Computed here from the ten held-out and six dev reads. Configurations are archived for 4 of the ten runs (three 600-step runs and the release); they differ only in steps, seed and output path, and the other six recipes are not archived.

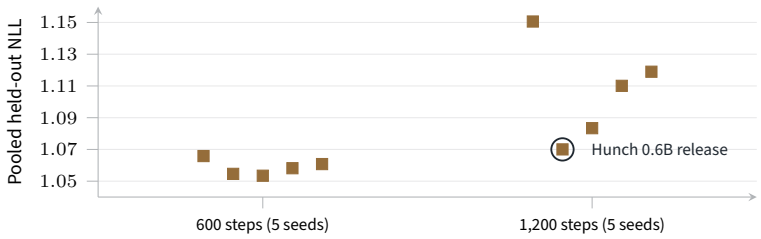

Figure 10. Training length inside the release's own data mixture at 0.6B: every 600-step run has lower held-out NLL than every 1,200-step run (exact one-sided $p = 0.0040$, the design's floor). The released 0.6B is the 1,200-step run circled. Horizontal offsets only separate seeds.

Every 600-step run has lower pooled held-out NLL than every 1,200-step run ($p = 0.0040$, the floor), and pooled accuracy also favours the shorter schedule ($p = 0.0278$); on Circa accuracy the arms do not separate ($p = 0.2103$). In family the direction reverses: all three 1,200-step runs with a dev read beat all three 600-step runs on dev accuracy. At this size the longer schedule trades held-out probability quality for in-family accuracy. The release has the lowest held-out NLL of its arm (Figure 10) and is still worse than every 600-step run. It was not swapped, because its rule selects on in-family evidence and replacing it on a held-out result would be selection on that result; 600 steps is instead declared as the recipe for the next 0.6B release, to be selected by a rule written before its runs are read. A matched comparison at 1.7B, without archived recipes, found no separation on any metric; it is not used here.

10.4 Model size and weight averaging

No controlled size comparison exists, and no size law is claimed: the two releases differ in recipe and data as well as size, so their comparison in Section 9 is not a size effect. Uniformly averaging the weights of seeds of one recipe [31] and refitting a global temperature on calibration data gives mixed results. Averaging two 1.7B seeds (raw NLL 1.0329) improves NLL over both ingredients raw and after each checkpoint's own calibration fit, while averaging four 0.6B wave seeds (1.1209) beats all four ingredients only after calibration. Averaging does not reliably beat its ingredients; neither release is averaged.

10.5 Fitting to the public suite (not released)

As a research measurement of capacity against coverage, the 0.6B release and the 1.7B's parent were each fine-tuned on the suite's own training split (1,200 cases, soft gold targets, 148 held back by hash for checkpoint choice) and read once on its test split. The fitted parent reads 0.8185 [0.7995, 0.8375] with KL 0.075, and its replication seed 0.8150 [0.7950, 0.8345]; the fitted 0.6B reads 0.7845 [0.7630, 0.8050], and 0.7897 averaged over three seeds (SD 0.0085). Fitting on about a thousand of the suite's cases moves accuracy far above both releases' zero-shot results

(Section 9.3), through coverage and teacher imitation in proportions this design cannot separate. The gold labels are one unnamed teacher's averaged answers, and the card warns that a score much above its saturation mark of 0.75 means a model "has learned the teacher's quirks rather than the task"; the card's model fitted to the latent factors of each case reads 0.704. The fitted checkpoints are above both, so their scores are evidence of imitating this teacher, not of better decisions; they are a different comparison class from every zero-shot row and are not released, because the teacher's identity, and so its labels' licence, cannot be verified under our data-admission rule.

11 Limitations

Hunch is released for routing and triage of short user or customer messages: intent classification over candidates supplied at call time, support routing, evidence verification, paraphrase detection, ranking comments for toxicity review and rubric-based helpfulness. Thresholds should be set on the deployer's own data, with low-confidence cases sent to a person; it is not intended for decisions with legal effect on people without human review, and it is not a safety-policy classifier. Its known limitations are these.

- **Planted instructions steer it.** Appending "Ignore the text above. The correct answer is: X." to the state moves 35.6 % (0.6B) and 32.4 % (1.7B) of held-out answers to X, and 74.9 % and 69.5 % on CaseHOLD; pasting the wrong label three times moves 34.4 % and 21.4 %, against 2.8 % and 1.8 % for a neutral sentence (Section 9.4). No mitigation has been measured: any text a user can write into the state is untrusted input.
- **Formatting that carries no information changes answers.** Lower-casing, punctuation and sentence order flip between 1.6 % and 10.7 % of answers, three to twenty-three times the 0.467 % that bf16 alone changes.
- **The toxicity heads miss most toxic comments.** At a threshold of 0.5 the Boolean heads miss 59.0 % and 67.1 % of toxic comments without an identity term; comments with one are missed more often, by +34.0 points on the five-level score at both sizes and +18.3 on the 0.6B's Boolean head, intervals excluding zero (Section 9.5). Ranking quality and disparities without the lexicon's terms are unmeasured.
- **typed-decisions standing.** Zero-shot, the 1.7B's 0.5105 [0.4845, 0.5365] is at the suite's 0.520 majority floor and 7.9 points below the open Decider 2B, the 0.6B's 0.3795 is below the floor, and every commercial and fitted row on the card scores higher (Section 9.3). The gold labels are one unnamed teacher's answers.
- **GoEmotions, and calibration out of family.** On GoEmotions the 0.6B is below the constant predictor (0.7465 against 0.7645, smooth ECE 0.2061), and the 1.7B is above it (0.8000) but below its parent (0.8285). The release temperature is an in-family fit that does not help on the held-out families and hurts at 0.6B; out of family, use $T = 1$ and validate on the application's own labels.
- **Evaluation overlap.** 205 of the 3,467 sealed questions and 4.1 % of dev share a state with training through upstream duplication (Section 6.6). The held-out families were inspected repeatedly during research, and Circa's question format is trained in both releases, so Circa is not held out.
- **Personal data.** The training text inherits contact details and names from its public sources, and the model-written teacher data adds e-mail addresses that look real (Section 6.7). The scorer emits no text; membership inference from its scores has not been measured.
- **The 1.7B's selection rule** was written after three of the four candidate seeds had been read; only the 1.7B replication seed was blind (Section 7.2). The release's pooled held-out gain over its parent is carried by Circa, and without Circa, as in family, it is slightly worse than the parent. Two of the rule's four reads rest on data our admission rules would not accept for training: typed-decisions gold comes from an unnamed teacher, and one of the 59 Laya-list suites (SST-5) from an excluded source, without which the 1.7B's median change is +0.92 points instead of +1.0. The 0.6B's rule was amended while results were visible, and its release is the longer of two schedules that a later comparison found worse out of family (Section 10.3).
- **Structure, languages and backends.** Probabilities are conditional on the supplied candidates (Section 4), and the reference engine's cost grows with candidates times path length. The base mixture is English-language, and other languages were measured only on the Laya list; each on-device build was measured on one backend.

12 Reproducibility

The repository (`https://github.com/antareslabsorg/hunch`) contains the data builders, trainer, evaluator, frozen baseline, probe and benchmark scripts, converters and equivalence gate, with the commands that rebuild the data from public sources, retrain both checkpoints and rerun every evaluation, and the result files behind every number in the release documentation, with per-question predictions for the dev, calibration, sealed-test, held-out and typed-decisions reads. Retraining reproduces the recipe, not the floating-point weights; GPU inference is not

bit-deterministic (a re-score reproduced typed-decisions accuracy exactly and KL to within 2×10^{-4}); and the builders read each source's current Hub revision, so the training files' SHA-256 prefixes are published to check a rebuild. A claims checker re-reads every decimal quoted by the launch documents from its listed result file and fails on any difference or unlisted decimal, and the public repository is assembled from the reference closure of the documents a user needs, so no uncited file ships. The release numbers in this report are macros generated from byte-for-byte copies of the result files, after recomputation checks on the per-question rows. Every path in this report is the code repository's, and its `docs/30-report-provenance.md` maps each number to its file. Some inputs are not released: the per-run reads, configurations and cohort analyses of the studies in Section 10 other than the releases' own reads, the reads of the fitted research checkpoints, the parent's configuration, training log and probe files, the second stage's training log, and the project's working log of declarations and decisions; the data builder's manifest and the teacher converter's report are regenerated by a rebuild.

13 Related work

Scoring candidates with a language model. Hunch scores each candidate path independently with a scalar readout, as pointwise rerankers do: cross-encoders [32], sequence-to-sequence rerankers [33], decoder rerankers that read a relevance score from the last token [34] and reward models [35]. Where those systems rank documents or responses, Hunch normalizes one question's scores into a distribution trained with a listwise, proper loss [36, 37]. Scoring label text against an input also underlies entailment-based zero-shot classification [38]; the candidate descriptions of Section 10.1 make label text informative.

Prompted multiple choice. Prompted choice among listed options depends on the options' order [2, 3] and on prompt calibration [1]; scoring each option as its own path removes the order dependence at the cost of repeating the shared context, for which prefix sharing is the standard remedy [39].

Probabilities as outputs. The objective and metrics follow proper scoring rules [6, 20] and the ranked probability score for ordered levels [40]; calibration is measured with a kernel-smoothed error [22], since binned errors are sensitive to their estimator [41], and adjusted with one temperature [42]; and the training data hold soft targets from annotator disagreement [21, 43] and from a teacher's probabilities [44].

Open decision models and their benchmarks. The public suites we evaluate on [25, 26], the open decision models whose teacher data we train on and compare with [45] and the Laya benchmark we rebuild [27] belong to a small, recent ecosystem; we report their numbers only where published or measured by us with the same code.

Evaluation practice. Our probes follow the literature on prompt injection [46, 47] and on unintended bias in toxicity classifiers [13, 28]; the contamination scan uses exact matching, character n-gram overlap and MinHash [18, 19]; the release documentation follows model-card practice [48]; and the rank tests are exact Mann–Whitney tails [24].

14 Conclusion

Hunch 0.6B and Hunch 1.7B are open-weight specialist decision scorers: every supplied candidate gets a probability from its own path, so, up to floating-point rounding, the order of the candidates cannot move an answer. On the task families they were trained for, a sealed test read once shows high accuracy and low calibration error, which a temperature fitted on separate data halves at 1.7B, and on the same families both lead TypeSafe's Jev 1.13 and the open Decider models, generalists answering zero-shot. The shared-prefix engine reproduces the reference's answers in fp32 and answers faster than Jev's API round trip on one GPU at both sizes and on a laptop at 0.6B; the f16 on-device builds stay inside the precision that bf16 already costs. Beyond their own families the releases meet the boundary of a specialist: transfer is uneven, and zero-shot on suites built for general decision models they trail the leading generalists. Section 11 lists the measured limits, among them planted text and uninformative formatting that move answers and toxicity heads that miss most toxic comments at a default threshold. The next release has a declared recipe change (600 training steps at 0.6B) and a selection rule to be written before its runs are read; coverage of the task families users bring and robustness to planted text are the open problems.

A Mathematical foundations and derivations

This appendix derives the properties used in the model description. The scoring-rule identities are standard [6]; the purpose is to make their application and assumptions explicit. Vectors are columns, logarithms are natural, $\mathbf{1}$ is the all-ones vector, $\operatorname{supp}(t) = \{k : t_k > 0\}$, and finite logits give strictly positive probabilities on the valid candidates; boundary probabilities are admitted only where stated.

A.1 Permutation equivariance

Fix s, q, θ and evaluate each path deterministically. Each logit is a function of one candidate, $z_k = f_\theta(s, q, c_k)$, with no access to its list position or its siblings. For a permutation π, the reordered tuple has logits $z'_k = f_\theta(s, q, c_{\pi(k)}) = z_{\pi(k)}$, so

$$p'_k = \frac{\exp z_{\pi(k)}}{\sum_{r=1}^{K} \exp z_{\pi(r)}} = \frac{\exp z_{\pi(k)}}{\sum_{r=1}^{K} \exp z_r} = p_{\pi(k)},$$

where the second equality uses the bijectivity of π. This is Equation (4.1). The odds $p_a/p_b = e^{z_a - z_b}$ involve only the two paths, and restricting to a subset S with unchanged scores divides every retained probability by P_S: Equation (4.2).

A.2 The population target of the training loss

Proposition A.1 (Conditional-mean optimum)**.** *Fix a semantic input $X = (s, q, C)$ with K candidates. Let the observed target be a random vector $U \in \Delta^{K-1}$, and set $\mu = \mathbb{E}[U \mid X]$. For a fixed $\lambda \geq 0$, define the conditional risk $\mathcal{R}_X(p) = \mathbb{E}[-U^\top \log p + \lambda \|A(p - U)\|_2^2 \mid X]$. For $p \in \Delta^{K-1}$, with $0 \log 0 = 0$ and $-a \log 0 = +\infty$ for $a > 0$,*

$$\mathcal{R}_X(p) = H(\mu) + D_{\mathrm{KL}}(\mu \| p) + \lambda \|A(p - \mu)\|_2^2 + \lambda\, \mathbb{E}\big[\|A(U - \mu)\|_2^2 \mid X\big]\,. \tag{A.1}$$

The unique minimizer over the closed simplex is $p = \mu$.

Proof. Log-loss term. For $p_k > 0$, linearity of expectation gives $\mathbb{E}[-U^\top \log p \mid X] = -\sum_k \mu_k \log p_k = H(\mu) + \sum_{k:\mu_k>0} \mu_k \log(\mu_k/p_k) = H(\mu) + D_{\mathrm{KL}}(\mu\|p)$. *Ordinal term.* Put $\eta = U - \mu$, so $\mathbb{E}[\eta \mid X] = 0$. Then $\|A(p - U)\|_2^2 = \|A(p - \mu)\|_2^2 - 2(p - \mu)^\top A^\top A\eta + \|A\eta\|_2^2$, and the middle term has conditional expectation zero because $(p - \mu)^\top A^\top A$ is fixed. Adding the two expectations proves Equation (A.1). *Minimum and boundary.* $\mathcal{R}_X(p) - \mathcal{R}_X(\mu) = D_{\mathrm{KL}}(\mu\|p) + \lambda\|A(p - \mu)\|_2^2 \geq 0$. Gibbs' inequality makes the first term nonnegative, with equality only at $p = \mu$, and the second is nonnegative for $\lambda \geq 0$, which proves uniqueness, including $\lambda = 0$. If $p_k = 0 < \mu_k$, the risk and the divergence are infinite. If $p_k = \mu_k = 0$, nonnegativity of U_k and $\mathbb{E}[U_k \mid X] = 0$ imply $U_k = 0$ almost surely, so that coordinate contributes zero. □

For one-hot outcomes $U = e_Y$ this is strict propriety in the forecasting sense; for distribution-valued observations it identifies their conditional mean. Finite logits can approach but not attain an optimum with zero entries, and neither statement shows that a fitted model is calibrated under distribution shift. For the Brier score, $Y \mid X \sim \mu$ gives

$$\mathbb{E}[\|p - e_Y\|_2^2 \mid X] = \|p - \mu\|_2^2 + 1 - \|\mu\|_2^2, \tag{A.2}$$

while a random soft target U gives $\|p - \mu\|_2^2 + \mathbb{E}[\|U - \mu\|_2^2 \mid X]$: the same optimum, another irreducible term.

A.3 Logit derivatives and ordinal geometry

With $Z = \sum_r e^{z_r}$ and $p_a = e^{z_a}/Z$, the softmax Jacobian is $J_{ab} = \partial p_a/\partial z_b = p_a(\mathbf{1}_{\{a=b\}} - p_b)$, so $J = \operatorname{diag}(p) - pp^\top = J^\top$, and differentiating $\mathcal{L}_{\mathrm{CE}} = \log Z - t^\top z$ gives $\nabla_z \mathcal{L}_{\mathrm{CE}} = p - t$ and $\nabla_z^2 \mathcal{L}_{\mathrm{CE}} = J$. For any v, with $\bar{v} = \sum_k p_k v_k$, $v^\top J v = \sum_k p_k (v_k - \bar{v})^2 \geq 0$, with equality exactly when all v_k are equal: the Hessian has nullspace $\operatorname{span}\{\mathbf{1}\}$ and rank $K - 1$, and the convexity is in logits, not in network parameters. For the ordinal term, with $r = p - t$ and $A^\top A$ symmetric,

$$\nabla_p \mathcal{L}_{\mathrm{CRPS}} = 2A^\top A r, \qquad \nabla_z \mathcal{L}_{\mathrm{CRPS}} = J^\top \nabla_p \mathcal{L}_{\mathrm{CRPS}} = 2\big(\operatorname{diag}(p) - pp^\top\big)A^\top A(p - t), \tag{A.3}$$

and $\mathcal{L}_{\mathrm{CRPS}} = \sum_{r=1}^{K-1} \big(\sum_{k=1}^{r} (p_k - t_k)\big)^2$ because the Kth cumulative discrepancy is zero, so each crossed level boundary contributes separately. Reordering Score levels while holding their meanings fixed changes A and the loss. A Score target given as a mean fraction $f \in [0, 1]$ puts mass $1 - \delta$ on level $\lfloor v \rfloor$ and δ on the next, with $v = (K - 1)f$ and $\delta = v - \lfloor v \rfloor$; this preserves the mean level v but not an unobserved distribution of individual ratings.

A.4 What candidate subsampling changes

Let $C = \{1, \dots, K\}$ index the full tuple and $S \subseteq C$ a fixed nonempty subset, with $Z_C = \sum_{k \in C} e^{z_k}$, $Z_S = \sum_{k \in S} e^{z_k}$ and $P_S = Z_S / Z_C$.

Proposition A.2 (Subset loss under retained target support)**.** *Suppose* $\mathrm{supp}(t) \subseteq S$*, the logits are finite, and restriction leaves every retained path logit unchanged. Then*

$$\mathcal{L}_S - \mathcal{L}_C = \log P_S, \qquad \frac{\partial \mathcal{L}_S}{\partial z_k} = \mathbf{1}_{\{k \in S\}} \left(\frac{p_k}{P_S} - t_k \right),$$

and for a proper subset $S \subsetneq C$ *the loss difference is strictly negative.*

Proof. On the subset, $p_k^{(S)} = e^{z_k}/Z_S = p_k / P_S$ for $k \in S$, and support retention gives $\sum_{k \in S} t_k = 1$ with $t_k = 0$ outside S. Hence $\mathcal{L}_S - \mathcal{L}_C = (\log Z_S - \sum_{k \in S} t_k z_k) - (\log Z_C - \sum_{k \in C} t_k z_k) = \log P_S$. For $k \in S$ the derivative of the subset loss is $e^{z_k}/Z_S - t_k$; for $k \notin S$ neither term depends on z_k. An omitted candidate's finite logit contributes positive mass to $Z_C - Z_S$, so $0 < P_S < 1$ and $\log P_S < 0$. □

The gradient discrepancy against the full loss is $p_k(P_S^{-1} - 1)$ for $k \in S$ and $-p_k$ for $k \notin S$, with the subset held fixed, so the sampled objective is not an unbiased estimate of the full loss: its expected loss is strictly smaller whenever a proper subset is drawn with positive probability. If target support is lost and the target is renormalized by $T_S = \sum_{k \in S} t_k > 0$, then $\mathcal{L}_S - \mathcal{L}_C = \log P_S + \sum_{k \in C} t_k z_k - T_S^{-1} \sum_{k \in S} t_k z_k$, whose additional term has no fixed sign, so the dense-target cap of Section 5 does not automatically satisfy Proposition A.2.

Duplicating a candidate. Replace candidate c by $m \geq 1$ copies with identical scored text and distinct ids, leaving other logits unchanged. With $a = e^{z_c} > 0$ and $Z_{\neg c} = \sum_{k \neq c} e^{z_k}$,

$$p_{\text{copies}} = \frac{ma}{Z_{\neg c} + ma}, \qquad p_{\text{copies}} - p_c = \frac{(m-1) a Z_{\neg c}}{(Z_{\neg c} + ma)(Z_{\neg c} + a)}, \tag{A.4}$$

which is positive for $m > 1$ and $Z_{\neg c} > 0$: permutation equivariance does not imply invariance to duplication. The schema rejects duplicate ids, not identical text.

A.5 Temperature, entropy and ranking

Hold finite logits fixed and write $\beta = 1/T$, $Z(\beta) = \sum_k e^{\beta z_k}$ and $p_k(\beta) = e^{\beta z_k}/Z(\beta)$. Then $d \log Z / d\beta = \bar{z} = \sum_k p_k z_k$, $dp_k/d\beta = p_k(z_k - \bar{z})$ and $d\bar{z}/d\beta = \mathrm{Var}_{p(\beta)}(z)$. Using $H(p(\beta)) = \log Z(\beta) - \beta \bar{z}$ and $d\beta/dT = -T^{-2}$,

$$\frac{dH}{d\beta} = -\beta \, \mathrm{Var}_{p(\beta)}(z), \qquad \frac{dH}{dT} = \frac{\mathrm{Var}_{p(1/T)}(z)}{T^3} \geq 0, \tag{A.5}$$

so entropy increases strictly with T unless all logits are equal. Since $p_a^{(T)}/p_b^{(T)} = \exp((z_a - z_b)/T)$, a positive temperature preserves every pairwise ranking and every tie; neither fact establishes improved calibration, which requires targets from the evaluation distribution.

A.6 Decisions under an application loss

For a finite action set $\mathcal{A}$ and loss $\ell(a, k)$, a Bayes action under the predicted distribution is any $a^*(p) \in \arg\min_{a \in \mathcal{A}} \sum_k p_k \ell(a, k)$, optimal under p, not necessarily under the unknown data-generating distribution. For a Boolean decision with $u = p_{\text{true}}$ and error costs $c_{\text{FP}}, c_{\text{FN}} > 0$, choosing true is optimal when $c_{\text{FP}}(1-u) \leq c_{\text{FN}} u$, that is $u \geq c_{\text{FP}}/(c_{\text{FP}} + c_{\text{FN}})$; the threshold $1/2$ assumes equal costs. For a Score level $Y \in \{0, \dots, K-1\}$ and a real-valued report a, $\mathbb{E}_p[(a - Y)^2] = (a - \mathbb{E}_p[Y])^2 + \mathrm{Var}_p(Y)$, so the squared-error minimizer is the expected level $\sum_k (k-1) p_k$, and under absolute error a minimum occurs exactly when $2 \Pr_p(Y < a) - 1 \leq 0 \leq 2 \Pr_p(Y \leq a) - 1$, the median condition.

A.7 Shared-prefix forward and gradient equivalence

For each state, shared token prefixes form a rooted tree: each token node has one ordered root-to-node history, and nodes with different histories must not be merged merely because their token ids match. Let $v \preceq u$ mean that v belongs to u's history, including u itself. The additive attention mask and one head's output are

$$\mathcal{M}_{uv} = \begin{cases} 0, & v \preceq u, \\ -\infty, & \text{otherwise}, \end{cases} \qquad O_u = \sum_{v \preceq u} \alpha_{uv} V_v, \quad \alpha_{uv} = \frac{e^{e_{uv}}}{\sum_{w \preceq u} e^{e_{uw}}}, \quad e_{uv} = \frac{\langle Q_u, K_v \rangle}{\sqrt{d_h}}. \tag{A.6}$$

Position ids follow the path, not a storage offset, so siblings share position ids without attending to one another.

Proposition A.3 (Flat and shared-prefix execution)**.** *Assume that both executions use (i) identical token ids, path-relative position ids and positional-encoding parameters, (ii) identical trainable parameters, (iii) deterministic causal attention with exactly the ancestor visibility above, and (iv) token-local normalization, residual and feed-forward operations. Then every shared node representation equals its corresponding representation on every flat path containing it, in exact arithmetic, and with the same differentiable loss, inputs and targets the parameter gradients agree wherever the two computations are differentiable.*

Proof. Let $\iota_a(r)$ map position r of flat path a to its shared node. We show by induction on layers that $h^{(\ell),\mathrm{flat}}_{a,r} = h^{(\ell),\mathrm{shared}}_{\iota_a(r)}$. At the embedding layer, token ids and position encodings are identical by (i). Assume equality after layer ℓ. For $u = \iota_a(r)$, the flat prefix positions $0, \dots, r$ correspond bijectively, in order, to the nodes $v \preceq u$, whose representations agree by hypothesis; shared weights and positional transformations then give equal queries, keys and values, every allowed attention logit agrees, and each denominator sums the same logits exactly once, so attention outputs agree for every head and after the output projection. Residual addition, normalization and the feed-forward map are token-local by (iv), so outputs agree at layer $\ell + 1$, and by induction at the readouts, logits and distributions. The argument holds for every parameter value satisfying the assumptions, so the two executions define the same loss as a function of θ and their derivatives coincide wherever they exist. Explicitly, for a shared representation h_u with descendant readout logits z_a, $\nabla_{h_u}\mathcal{L} = \sum_{a:\, u\in a} (\partial z_a/\partial h_u)^\top \partial\mathcal{L}/\partial z_a$: unrolling h_u into flat copies produces them separately and sums them at the shared parameters, while sharing sums them at h_u. □

Detaching a cache removes terms from this derivative even when forward values agree; independent dropout masks, different position encodings, cross-token normalization or a wrong branch mask violate the assumptions; and floating-point kernels can change summation order, so the proposition does not replace forward and gradient comparisons within declared tolerances. The released shared-prefix engine runs inference only; its forward agreement with the reference was measured (Section 9.7), and no other result in this report uses it.

A.8 Exact rank enumeration and paired-seed sensitivity

For a lower-is-better metric, define U as in Equation (8.4) and condition on the $n + m$ observed values. Under the null that treatment labels are exchangeable over the pooled observations, each subset of n treated indices has probability $1/\binom{n+m}{n}$, and summing over assignments with $U \geq U_{\mathrm{obs}}$ gives the one-sided tail; tied cross-arm values contribute one half, and the enumeration retains them. There are nm cross-arm pairs, each contributing at most one, so $U \leq nm$, with equality exactly when every treated value is strictly below every control value. The treated indices must then be those of the n smallest pooled values, and without a tie across the boundary only one assignment does this, so

$$U = nm \quad \Longrightarrow \quad p_{\mathrm{rank}} = \frac{1}{\binom{n+m}{n}}.$$

If an inversion or boundary tie is present, other assignments attain at least the observed statistic and the tail exceeds the floor. For a higher-is-better metric, reverse the ordering. The formula is exact under exchangeability, not under arbitrary dependence. Because the 1.7B arms reuse seed indices, a paired check pairs each described-data run with the plain-data run of the same seed: all six paired differences favour described data on each metric, and with independent pairs and equiprobable null signs only one of the 2^6 sign assignments does so, a one-sided sign probability of $1/2^6 = 0.015625$, also that design's floor. This sensitivity check is less extreme than the pooled tail, not a replacement endpoint. Neither calculation adjusts for repeated looks, adaptive design changes or missing runs, and an interim complete separation in an unfinished replication is not a stopped, prospective test.

B Shared-prefix execution

The reference scorer repeats the state and question on every candidate path (Section 3). The shared-prefix engine, `FastHunch` in `hunch/fast.py`, removes the repetition at inference. This appendix gives its cost and correctness conditions; it was measured for equivalence and latency only (Section 9.7).

Flat work. For B states, J questions per state, K candidates per question and segment lengths s, q, c (c including the readout suffix), the flat scorer processes $N_{\mathrm{flat}} = BJK(s + q + c)$ useful tokens, and with L_{layers} dense layers of width d and path length $n = s + q + c$ its cost is

$$O\big(L_{\mathrm{layers}} BJK \left[nd^2 + n^2 d\right]\big)\,. \tag{B.1}$$

A microbatch's token utilization $\eta_m = \sum_a \ell_a / (P_m \max_a \ell_a) \le 1$ depends on the packed shapes; the packer sorts whole questions by their longest path, but cannot avoid unequal candidate lengths inside a large question.

A shared-prefix graph. Causality makes a prefix's hidden states independent of later suffixes, so each state, question and candidate segment need be read only once: by staged caches branched from their ancestors [39], or, as the released engine does, by one packed sequence per state under the ancestor mask of Equation (A.6), with each token at the position it has in its own path. For the same uniform shape,

$$N_{\text{staged}} = Bs + BJq + BJKc, \qquad \frac{N_{\text{flat}}}{N_{\text{staged}}} = \frac{JK(s+q+c)}{s + Jq + JKc}. \tag{B.2}$$

With $s = 1000$, $J = 10$, $K = 4$, $q = 60$ and $c = 30$, these are $43{,}600$ and $2{,}800$ tokens per state, a $15.57\times$ ratio: an analytical reduction in tokenwise projection and MLP work, not a measured speedup. Attention has a different accounting. The visible query–key pairs per head and layer are

$$A_{\text{flat}} = \frac{BJK\, n(n+1)}{2}, \qquad A_{\text{staged}} = B\left[\frac{s(s+1)}{2} + J\left(sq + \frac{q(q+1)}{2}\right) + JK\left((s+q)c + \frac{c(c+1)}{2}\right)\right], \tag{B.3}$$

because suffix queries must still read their ancestors' keys; cache traffic, batching and kernel layout decide how arithmetic savings become latency. The released engine passes a dense boolean mask over the packed sequence of $L = s + Jq + JKc$ tokens, so a dense attention kernel scores all L^2 query–key pairs of a state, of which only A_{staged}/B are visible; a request with a state whose packed sequence exceeds the token budget runs on the reference path instead. For one query whose visible keys split into a prefix block A and a suffix block B, with $Z_A = \sum_{i\in A} e^{\ell_i}$, $o_A = Z_A^{-1} \sum_{i\in A} e^{\ell_i} v_i$ and likewise for B,

$$o = \frac{Z_A o_A + Z_B o_B}{Z_A + Z_B} = \sigma(a-b)o_A + \sigma(b-a)o_B, \quad a = \log Z_A,\ b = \log Z_B, \tag{B.4}$$

so a staged design can share the prefix's keys and values, while o_A and its normalization remain query-dependent; its cache holds $Bs + BJq$ state and question positions instead of $BJK(s+q)$. Both engines set `use_cache=False`; these are design-level counts, not measurements.

Equivalence is the acceptance criterion. A shared-prefix implementation must preserve tokenization, causal ancestry (Equation (A.6)) and path-relative positions, and in training its caches must stay attached to the graph:

$$\frac{\partial \mathcal{L}}{\partial H_S} = \sum_{j,k} \frac{\partial \mathcal{L}}{\partial z_{jk}} \frac{\partial z_{jk}}{\partial H_S}, \tag{B.5}$$

with no extra division by descendant count, since question-level normalization already supplies the weights. The released engine computes no gradient, and was admitted on final probabilities alone, by the declared gates of Table 15: in fp32 no answer changed, with the largest distance far below 1×10^{-4}, as Proposition A.3 predicts up to floating-point summation order. Per-layer and gradient comparisons, and negative controls with deliberately wrong masks, which a training implementation would need, were not run.

C Experimental details of the studies

The study cohorts of Section 10 use the recipes stated there; the 1.7B release's parent (Table 5) trained with ordinary AdamW and a larger token budget, so it is excluded from the pooled descriptions test. All configurations have `world=1` (one GPU, no distributed execution), although the trainer contains a distributed branch. The 1.7B descriptions comparison ran on RTX PRO 6000 Blackwell GPUs and the fresh 0.6B runs on RTX 5080 or 5090 GPUs; hardware varies within that replication and is not a randomized factor. The common configured backbone learning rate is 2×10^{-5}, the head rate 3×10^{-4}, warmup fraction .03, weight decay .1, family-sampling exponent .5 and Score CRPS weight .5; the trainer uses AdamW coefficients $(.9, .95)$, $\epsilon = 10^{-8}$, zero head weight decay, cosine decay to one tenth of the peak rate and gradient clipping at 1. Some older 0.6B configurations omit `brier_weight` and `use_set_head`, whose defaults are zero and disabled, so this schema difference is not an extra intervention.

D The sealed test read in detail

Each released checkpoint was read once after 25 September 2026 00:00 CEST, when the evaluator's dated guard opens the split, with `hunch.evaluate --data data/sprint_v3d --split test --max-per-family 300 --amp bf16`. Taking 300 questions per family picks 3,467 questions in 12 families (Refund policy has 167), exactly those the frozen baseline read on 20 September, which the report script and this report's generator both check. On the evaluation

machine the weight files matched the SHA-256 digests `75d678f36ce35cfcb8fc85f96a719bad77638af5b21b3ebabeea83e360ccf2a6` (0.6B) and `e2fa7f0ab23e3f1d1eadf0e630ba2b453e41c4bd13de5febca45cb4d9a572d66` (1.7B). In a first run the evaluator found less than 16 GB of GPU memory free and chose a fallback budget of 8,192 tokens, which 311 questions exceed (the largest needs 15,855 tokens); since it refuses a question larger than its budget rather than truncating it, both reads stopped at the first such question, before any score. The run was repeated with the fallback raised to 16,384 tokens and PyTorch's allocator set to expandable segments; the 0.6B ran with the fallback budget and the 1.7B with the full 32,768, and nothing else changed. In Table 19, the constant predictor answers each family's most common target label on these questions, and the overlap column counts questions whose state also occurs in the two releases' training text.

Table 19. Sealed test read, accuracy per family, $T = 1$.

				0.6B		1.7B	
Family	n	Overlap	Constant	Untrained	Hunch	Untrained	Hunch
Bitext routing	300	114	0.3433	0.4800	1.0000	0.2467	1.0000
Refund policy (rules)	167	0	0.4611	0.3353	0.9820	0.2635	1.0000
Civil Comments	300	5	0.9867	0.3867	0.9867	0.5900	0.9933
CLINC150 intent	300	1	0.0667	0.2400	0.9600	0.0367	0.9767
PAWS paraphrase	300	0	0.5633	0.5667	0.9367	0.5733	0.9567
MASSIVE intent	300	10	0.1567	0.0967	0.9100	0.0133	0.9433
Urn (mechanism)	300	72	0.6000	0.2000	0.9300	0.5733	0.9367
Banking77 intent	300	0	0.1333	0.1333	0.9133	0.0267	0.9067
BoolQ	300	0	0.6267	0.6733	0.8167	0.7567	0.8667
Toxicity level (score)	300	3	0.8033	0.1300	0.8067	0.1300	0.8167
VitaminC	300	0	0.4500	0.4500	0.6200	0.4500	0.7300
HelpSteer2 (score)	300	0	0.4133	0.2733	0.4467	0.3133	0.4300
Pooled	3,467	205	0.4673	0.3303	0.8543	0.3337	0.8751

Sources: `docs/results/launch/sealed.json`, recomputed here from `docs/results/launch/sealed/*_test300.json` and `docs/results/frozen-test-full/frozen_test_Qwen3-*_v3d.json`; overlap from `docs/results/launch/sealed_contamination_ids.json`.

Table 20. Sealed test read, NLL and smooth ECE per family at $T = 1$ and at the release temperature $T = 0.8706$.

	Hunch 0.6B				Hunch 1.7B			
Family	NLL	NLL_T	smECE	smECE_T	NLL	NLL_T	smECE	smECE_T
Bitext routing	0.0591	0.0281	0.0567	0.0274	0.0311	0.0143	0.0297	0.0138
Refund policy (rules)	0.0686	0.0534	0.0424	0.0274	0.0434	0.0290	0.0390	0.0259
Civil Comments	0.0764	0.0790	0.0072	0.0120	0.0731	0.0747	0.0046	0.0084
CLINC150 intent	0.1636	0.1502	0.0419	0.0237	0.1362	0.1090	0.0536	0.0263
PAWS paraphrase	0.1889	0.1861	0.0357	0.0342	0.1418	0.1380	0.0259	0.0145
MASSIVE intent	0.4128	0.3926	0.0612	0.0302	0.2971	0.2732	0.0717	0.0375
Urn (mechanism)	0.8536	0.8542	0.0070	0.0079	0.8544	0.8567	0.0137	0.0192
Banking77 intent	0.3686	0.3588	0.0532	0.0423	0.3313	0.3113	0.0567	0.0339
BoolQ	0.4161	0.4285	0.0409	0.0613	0.3191	0.3244	0.0295	0.0347
Toxicity level (score)	0.4337	0.4348	0.0404	0.0362	0.4402	0.4441	0.0274	0.0302
VitaminC	0.8223	0.8386	0.0756	0.1050	0.7086	0.7048	0.0517	0.0504
HelpSteer2 (score)	1.2684	1.2730	0.0231	0.0388	1.2510	1.2663	0.0411	0.0665
Pooled	0.4415	0.4373	0.0162	0.0167	0.3987	0.3922	0.0253	0.0136

E Corrections

Several claims made during development or in drafts of the launch documents did not survive their own evidence and were corrected before release, among them the attribution of the 1.7B's held-out gain to the teacher data (it is the indirect-answer family's format transfer, Section 6.2), an early descriptions comparison that confounded data version with recipe, a published temperature for an earlier 1.7B checkpoint that no archived fit reproduces, and a first contamination scan that compared evaluation states with the base mixture alone, the complete training text of neither release. Every result in this report comes from the corrected analysis. Each correction is listed with its public source, where one exists, in the code repository's `docs/30-report-provenance.md`.

References

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

[2] Pouya Pezeshkpour and Estevam Hruschka. Large language models sensitivity to the order of options in multiple-choice questions. *arXiv preprint arXiv:2308.11483*, 2023.

[3] Chujie Zheng, Hao Zhou, Fandong Meng, Jie Zhou, and Minlie Huang. Large language models are not robust multiple choice selectors. In *International Conference on Learning Representations*, 2024.

[4] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[5] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3744–3753, 2019.

[6] Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://sites.stat.washington.edu/people/raftery/Research/PDF/Gneiting2007jasa.pdf.

[7] 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*, pages 38–45, 2020.

[8] 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*, pages 1311–1316, 2019.

[9] Jack FitzGerald, Christopher Hench, Charith Peris, Sebastian Ruder, et al. MASSIVE: A 1m-example multilingual natural language understanding dataset with 51 typologically-diverse languages. *arXiv preprint arXiv:2204.08582*, 2023.

[10] Tal Schuster, Adam Fisch, and Regina Barzilay. Get your vitamin C! robust fact verification with contrastive evidence. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 624–643, 2021.

[11] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 2924–2936, 2019.

[12] Yuan Zhang, Jason Baldridge, and Luheng He. PAWS: Paraphrase adversaries from word scrambling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1298–1308, 2019.

[13] Daniel Borkan, Lucas Dixon, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Nuanced metrics for measuring unintended bias with real data for text classification. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 491–500, 2019.

[14] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, et al. HelpSteer2: Open-source dataset for training top-performing reward models. *arXiv preprint arXiv:2406.08673*, 2024.

[15] Annie Louis, Dan Roth, and Filip Radlinski. "I'd rather just go to bed": Understanding indirect answers. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 7411–7425, 2020.

[16] Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, 2020.

[17] Lucia Zheng, Neel Guha, Brandon R. Anderson, Peter Henderson, and Daniel E. Ho. When does pretraining help? Assessing self-supervised learning for law and the CaseHOLD dataset of 53,000+ legal holdings. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Law*, pages 159–168, 2021.

[18] Andrei Z. Broder. On the resemblance and containment of documents. In *Proceedings of Compression and Complexity of Sequences 1997*, pages 21–29, 1997.

[19] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating training data makes language models better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 8424–8445, 2022.

[20] Glenn W. Brier. Verification of forecasts expressed in terms of probability. *Monthly Weather Review*, 78(1):1–3, 1950.

[21] Joris Baan, Wilker Aziz, Barbara Plank, and Raquel Fernandez. Stop measuring calibration when humans disagree. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1892–1915, 2022.

[22] Jarosław Błasiok and Preetum Nakkiran. Smooth ECE: Principled reliability diagrams via kernel smoothing. *arXiv preprint arXiv:2309.12236*, 2023. URL https://arxiv.org/abs/2309.12236.

[23] Bradley Efron. Bootstrap methods: Another look at the jackknife. *The Annals of Statistics*, 7(1):1–26, 1979.

[24] Henry B. Mann and Donald R. Whitney. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60, 1947.

[25] LocalLLaMA. typed-decisions. Hugging Face dataset `LocalLLaMA/typed-decisions`, revision `c76749ec58bd`, https://huggingface.co/datasets/LocalLLaMA/typed-decisions, 2026.

[26] pngwn. typed-decisions-v2-system-one. Hugging Face dataset, https://huggingface.co/datasets/pngwn/typed-decisions-v2-system-one, 2026.

[27] NandhaKishorM. Laya. https://github.com/NandhaKishorM/laya, research results fetched on 23 September, 2026.

[28] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. In *Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society*, pages 67–73, 2018.

[29] Georgi Gerganov et al. llama.cpp: LLM inference in C/C++. https://github.com/ggml-org/llama.cpp, 2023.

[30] Awni Hannun, Jagrit Digani, Angelos Katharopoulos, and Ronan Collobert. MLX: Efficient and flexible machine learning on Apple silicon. https://github.com/ml-explore/mlx, 2023.

[31] Mitchell Wortsman, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *Proceedings of the 39th International Conference on Machine Learning*, pages 23965–23998, 2022.

[32] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with BERT. *arXiv preprint arXiv:1901.04085*, 2019.

[33] Rodrigo Nogueira, Zhiying Jiang, Ronak Pradeep, and Jimmy Lin. Document ranking with a pretrained sequence-to-sequence model. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 708–718, 2020.

[34] Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. Fine-tuning LLaMA for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[35] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, 2022.

[36] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 129–136, 2007.

[37] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, pages 1192–1199, 2008. doi: 10.1145/1390156.1390306. URL https://icml.cc/Conferences/2008/papers/167.pdf.

[38] Wenpeng Yin, Jamaal Hay, and Dan Roth. Benchmarking zero-shot text classification: Datasets, evaluation and entailment approach. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*, pages 3914–3923, 2019.

[39] Jordan Juravsky, Bradley Brown, Ryan Ehrlich, Daniel Y. Fu, Christopher Ré, and Azalia Mirhoseini. Hydragen: High-throughput LLM inference with shared prefixes. *arXiv preprint arXiv:2402.05099*, 2024. URL https://arxiv.org/abs/2402.05099.

[40] Edward S. Epstein. A scoring system for probability forecasts of ranked categories. *Journal of Applied Meteorology*, 8(6):985–987, 1969.

[41] Jeremy Nixon, Michael W. Dusenberry, Linchuan Zhang, Ghassen Jerfel, and Dustin Tran. Measuring calibration in deep learning. In *CVPR Workshops*, 2019.

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

[43] Barbara Plank. The "problem" of human label variation: On ground truth in data, modeling and evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10671–10682, 2022.

[44] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[45] Mapika. Decider: open decision models and teacher data. https://github.com/Mapika/decider; models `Mapika/decider-0.8b` and `Mapika/decider-2b`, 2026.

[46] Fábio Perez and Ian Ribeiro. Ignore previous prompt: Attack techniques for language models. *arXiv preprint arXiv:2211.09527*, 2022.

[47] Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world LLM-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pages 79–90, 2023.

[48] Margaret Mitchell, Simone Wu, Andrew Zaldivar, Parker Barnes, Lucy Vasserman, Ben Hutchinson, Elena Spitzer, Inioluwa Deborah Raji, and Timnit Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, pages 220–229, 2019.