

Jeva: System One Models from First Principles

Qi Li Xinchao Wang

National University of Singapore
liqi@u.nus.edu; xinchao@nus.edu.sg

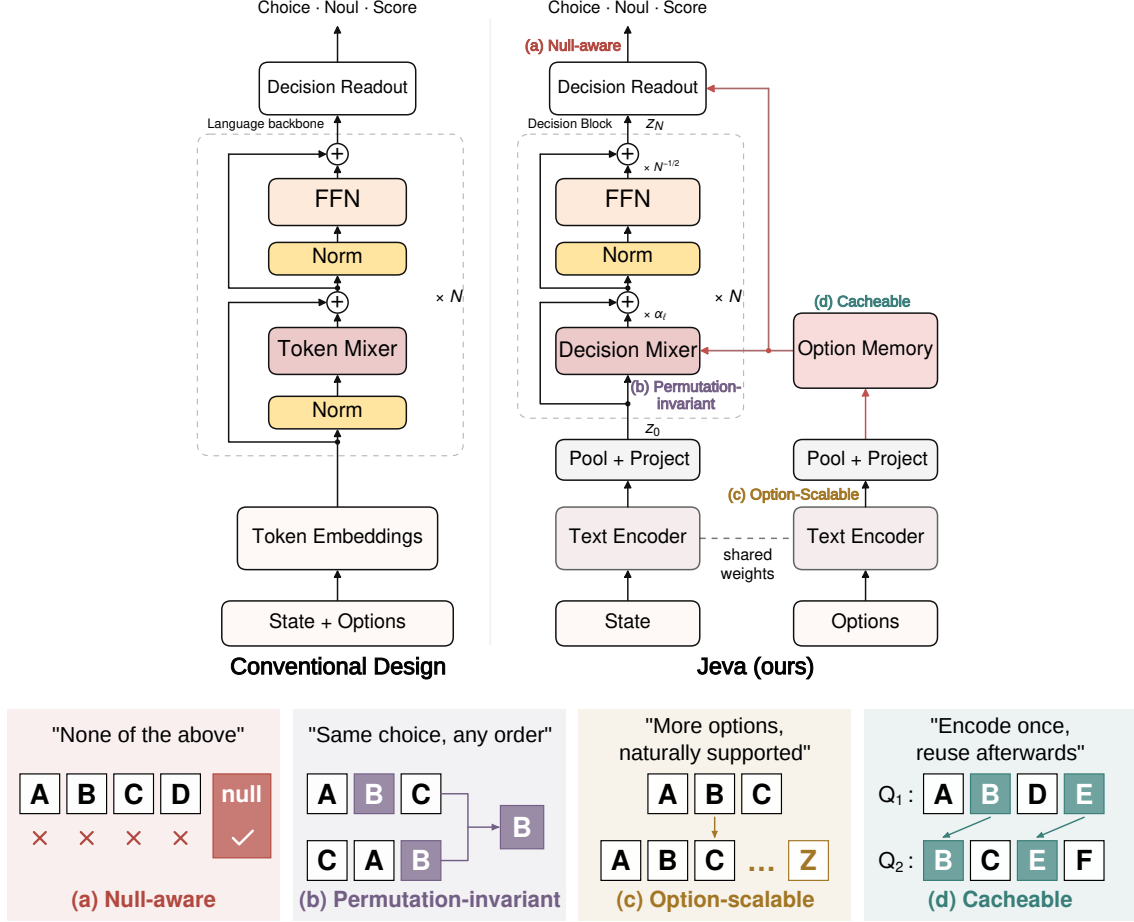


Figure 1: **Top**: a representative joint-encoding system one architecture and Jeva; N denotes the number of blocks in each architecture. **Bottom**: schematic illustrations of four structural properties: (a) an explicit null outcome when no candidate applies; (b) unchanged choices under candidate reordering; (c) support for variable-size candidate sets; and (d) reuse of option representations across requests.

Abstract

System one models enable direct, low-latency decision-making. Yet recent practice of predicting an answer from serialized text does not, by itself, respect the structure of a decision problem. We argue that current designs underemphasize four important structural requirements. **(i) null-aware**: Models should distinguish answer absence from uncertainty among candidates; **(ii) permutation-invariant**: Reordering alternatives should not change the decision; **(iii) option-scalable**: Growing candidate sets should not require redefining the output space; **(iv) cacheable**: Request-independent computation over recurring alternatives should be reusable across requests. We introduce **Jeva**, a decision-native architecture that addresses these requirements jointly. A shared encoder independently represents the request and each option, while a compact decision stack refines only the request state against immutable option memory. Symmetric aggregation yields permutation-equivariant probabilities; shared scoring accommodates variable-size candidate sets. A separate coverage estimate distinguishes answer absence from candidate competition, enabling explicit null outcomes. Request-independent option representations and layer projections support reuse across newly assembled sets. These mechanisms unify Choice, Noul, and Score outputs. Across evaluations, Jeva combines strong typed-decision performance with near-perfect null detection and no decision flips under any tested permutations. It scales to thousands of candidates and substantially reduces latency through cross-request reuse without changing decisions.

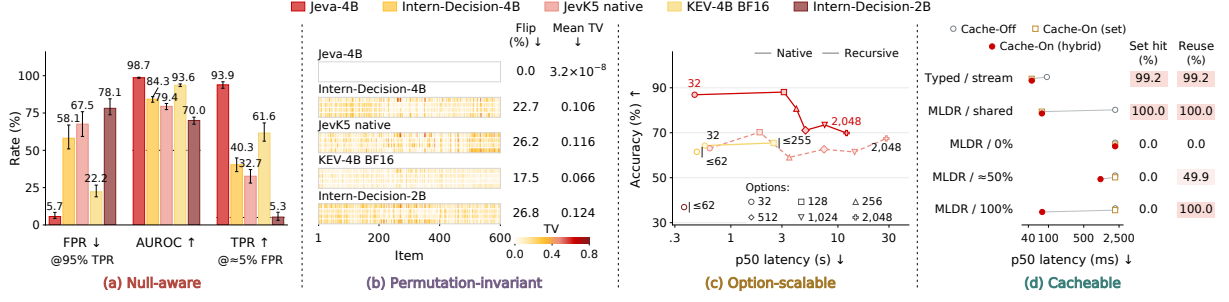


Figure 2: **Four structural properties of Jeva.** (a) Null detection on BANKING77 (Casanueva et al., 2020) with 16 candidates. (b) Candidate-aligned probability changes under Choice permutations from Typed Decisions (Locall-LaMA, 2026), with flip rates and mean total variation (TV). (c) Accuracy–latency trade-offs as candidate counts increase on our ToolRet-based diagnostic (Shi et al., 2025); solid and dashed lines denote native and recursive selection, respectively. (d) Cache-Off and Cache-On (set-cache, hybrid-cache) latency on Typed Decisions (Locall-LaMA, 2026) and MLDR (Chen et al., 2024), with set-hit rates and effective reuse.

1 Introduction

Many language-model requests require a compact decision: choose a tool, identify a document, test a condition, or assign a score (Schick et al., 2023; Qin et al., 2024; Saeidi et al., 2018; Zheng et al., 2023). System one models address these requests directly through typed outputs rather than generated rationales (Almeida, 2026). Such tasks impose structural requirements on the decision process. In candidate selection, reordering the alternatives should preserve the decision, and varying the candidate count should not require a new output space. The model must also recognize when none of the available alternatives applies. Meanwhile, a tool registry, document collection, or label vocabulary can persist across changing requests, creating opportunities to reuse computation and reduce decision latency. Together, these requirements call for a decision-native architecture.

However, many open-source system one models jointly encode requests and candidates as text, then read decisions from answer-token logits or a task-specific prediction head (JevK5 authors, 2026; InternLM, 2026; Lee, 2026). This design places reusable candidate content inside request-specific computation: changing the request or recombining candidates requires processing their joint context again. It also treats an unordered decision set as an ordered sequence, allowing incidental candidate positions to influence predictions. Implementations based on a fixed inventory of answer symbols further restrict candidate counts, while normalizing scores over the supplied candidates does not establish whether any candidate is acceptable. These limitations point to a common architectural issue: semantic representation, candidate interaction, and

answer availability are coupled within the same prediction process. A decision-native architecture should separate these roles, preserving context-sensitive judgment while making candidate reuse, order symmetry, and variable-size choice explicit properties of its computation.

We introduce **Jeva**, a decision-native architecture that separates reusable semantic encoding from set-dependent judgment. In Jeva, as shown in Figure 1, a shared language encoder independently maps the request and each option into compact vectors. A lightweight decision module then refines only the request state by reading an immutable option memory. This lets the state adapt to the candidate set while keeping option representations request-independent. Even a single decision block can reverse the preference between two retained options when the surrounding set changes, capturing interactions that independent candidate scoring cannot express. Symmetric aggregation in the decision mixer and shared candidate scoring in the readout yield permutation-equivariant option probabilities and accommodate variable-size candidate sets without expanding the output parameters. A separate coverage gate allocates probability between null and the candidate set, distinguishing answer absence from uncertainty among alternatives; a conditional distribution resolves competition within the set. Finally, both option vectors and their layer-specific projections can be cached across requests, including newly assembled candidate sets, while state refinement and decision readout remain specific to each request.

We substantiate these design principles through theoretical analysis and empirical evaluation against specialized system one models. Our analysis establishes single-block set-dependent expres-

sivity, permutation-equivariant probabilities, and exact reuse of option-side computation, while distinguishing coverage estimation from candidate-score normalization. With a Qwen3.5-4B backbone and one decision block, Jeva combines strong typed-decision performance with nearly perfect null detection and stable choices under all tested candidate permutations. These benefits extend beyond small candidate sets: at 2,048 options, Jeva’s native full-set scoring achieves higher accuracy and a speedup of over $2\times$ in median latency over the recursive-selection baseline. Cross-request reuse provides a complementary efficiency gain, delivering a median-latency speedup of over $20\times$ for newly assembled long-document candidate sets without changing decisions. Together, these findings (Figure 2) demonstrate how a decision-native architecture unifies null awareness, permutation invariance, option scalability, and cacheability.

2 Related Work

Direct and typed decision models. Open-source system one models produce structured decisions through several readout designs. SemIf and JevK5 extract option probabilities from answer-token logits without autoregressive decoding (Lee, 2026; JevK5 authors, 2026), while Intern-Decision uses decision markers to read distributions over permitted answer symbols (InternLM, 2026). ReFlex aggregates predictions across different option orders to reduce sensitivity to candidate position (Raghavan, 2026). Beyond token-logit readouts, KEV uses a pointer head over contextualized option positions (Palmer, 2026), and Laya combines a bidirectional language encoder with a dedicated decision head (Convai Innovations, 2026). These methods derive decisions from request and candidate token representations. Jeva separates independent text encoding from compact, set-conditioned state updates, making candidate computation reusable while retaining interactions among alternatives.

Independent encoding and late interaction. Separating query and candidate encoding enables reusable representations and efficient matching. Dense Passage Retrieval encodes questions and passages independently, allowing corpus representations to be precomputed (Karpukhin et al., 2020). ColBERT retains token-level representations to support fine-grained late interaction (Khattab and Zaharia, 2020). Contrastive Language Models extend independent encoding and candidate caching to

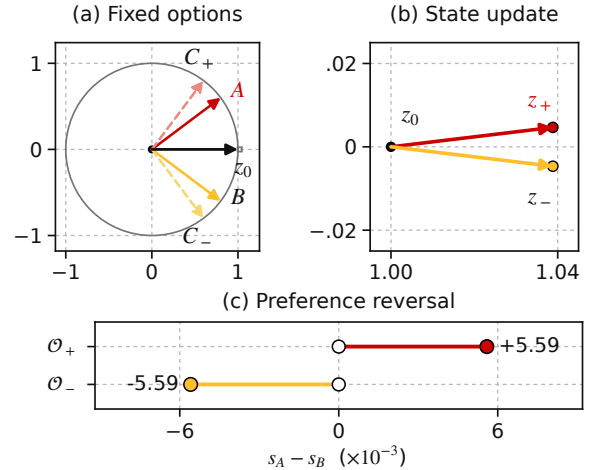


Figure 3: **Fixed options, set-dependent decisions.** Replacing C_+ by C_- changes the refined state and reverses the A – B margin. Hollow markers show the independent-score tie. The state zoom uses equal scales; values follow Proposition 1.

typed Choice, Noul, and Score decisions (Kwok et al., 2026). Jeva refines the request state against the full candidate set before scoring and coverage estimation, enabling set-dependent preferences while keeping candidate vectors and layer projections reusable.

Set functions and latent attention. Deep Sets establishes shared transformations and symmetric aggregation as a foundation for permutation-invariant set modeling (Zaheer et al., 2017). Set Transformer introduces attention-based interactions among set elements (Lee et al., 2019), while Perceiver processes large inputs through a compact latent representation (Jaegle et al., 2021). Jeva refines a request-conditioned state against immutable option memory, combining set-dependent judgment with reusable representations. Symmetric aggregation and shared scoring yield permutation-equivariant probabilities and support variable-size sets without changing parameter shapes. At fixed width and depth, refinement and readout scale linearly with candidate count over prepared memory.

Null awareness and calibration. Answerability and abstention address related but distinct questions. SQuAD 2.0 evaluates answerability (Rajpurkar et al., 2018), while selective classification permits abstention to control prediction risk (Geifman and El-Yaniv, 2017). Jeva defines null as the absence of a valid candidate. Its coverage gate distinguishes the absence of a valid candidate from uncertainty among alternatives. Post-training temperature scaling provides complementary probability calibration (Guo et al., 2017).

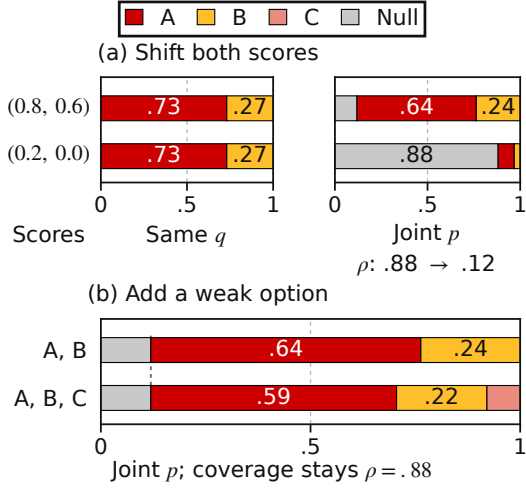


Figure 4: **Competition differs from coverage.** (a) A score shift preserves q but changes ρ . (b) Adding $s_C = 0.4$ at fixed scores preserves ρ but redistributes p . Here $\tau_{\text{dec}} = 0.20$, $\tau_{\text{cov}} = 0.15$, and $b_{\text{cov}} = 0.50$.

3 Jeva: A Native Decision Architecture

Jeva independently encodes the request and each option with a shared backbone (Figure 1). A compact decision module refines only the request state against immutable option memory, enabling reuse while retaining set-dependent scoring.

3.1 Independent Encoding, Collective Decisions

A request comprises state x , question ι , decision type t , and candidates $\mathcal{O} = (o_1, \dots, o_K)$, with $K \geq 2$ in the typed interface. We format inputs with two text templates: $T_x(t, \iota, x)$ combines the decision type, question, and state, while $T_o(t, o_i)$ contains only the decision type and candidate text. Let E_θ be the language backbone followed by last-nonpadding-token pooling. Bias-free projections give

$$\begin{aligned} z_0 &= P_x E_\theta(T_x(t, \iota, x)), \\ u_i &= \text{nrm}(P_o E_\theta(T_o(t, o_i))), \end{aligned} \quad (1)$$

where nrm is stabilized ℓ_2 normalization and $z_0, u_i \in \mathbb{R}^d$. Templates identify each input’s type and role; T_o contains no current question, state, candidate index, or other option. Each text is contextualized by the backbone before pooling. The rows of $U = [u_1^\top; \dots; u_K^\top]$ stay fixed within a forward pass, while the encoder and projections remain trainable.

Why refine the state? Let q denote the distribution over candidates conditional on an acceptable candidate being present. For independent scores $g(x, \iota, o_i)$ normalized by softmax with temperature

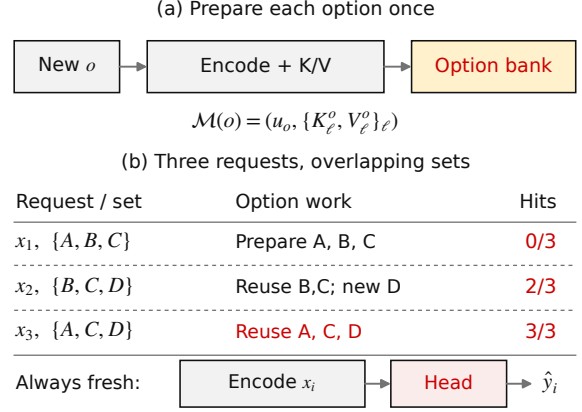


Figure 5: **Reuse across new requests and sets.** With a fixed model, type, and encoding policy, an initially empty option bank gives 0/3, 2/3, and 3/3 hits without eviction. Every whole-set lookup misses; state encoding and decisions remain fresh.

$\tau > 0$, its log odds satisfy

$$\log \frac{q_i(\mathcal{O})}{q_j(\mathcal{O})} = \frac{g(x, \iota, o_i) - g(x, \iota, o_j)}{\tau}. \quad (2)$$

The shared denominator cancels in the ratio: adding or removing other candidates can change probabilities but cannot reverse the preference between two retained options. Jeva instead conditions the scoring state on the candidate set, enabling such reversals through compact per-request computation while keeping option representations reusable.

3.2 Candidate-Conditioned State Refinement

Each of L blocks reads U through H heads. For head h , define $q_\ell^h = \text{nrm}([W_\ell^Q z_\ell]_h)$, $k_{\ell i}^h = \text{nrm}([W_\ell^K u_i]_h)$, and $v_{\ell i}^h = [W_\ell^V u_i]_h$. The candidate-weighted message is

$$\begin{aligned} a_{\ell i}^h &= \frac{\exp((q_\ell^h)^\top k_{\ell i}^h / \tau_\ell)}{\sum_j \exp((q_\ell^h)^\top k_{\ell j}^h / \tau_\ell)}, \\ m_\ell &= W_\ell^O \text{Concat}_{h=1}^H \left(\sum_i a_{\ell i}^h v_{\ell i}^h \right). \end{aligned} \quad (3)$$

Padding is masked. For compatibility logits $c_i = (q_\ell^h)^\top k_{\ell i}^h$, these weights uniquely maximize $a^\top c + \tau_\ell \mathcal{H}(a)$ over the probability simplex, yielding an entropy-regularized allocation of candidate evidence for state refinement. A derivation is provided in Appendix A.1. The state undergoes two residual updates:

$$\begin{aligned} \tilde{z}_\ell &= z_\ell + \alpha_\ell m_\ell, \\ z_{\ell+1} &= \tilde{z}_\ell + L^{-1/2} F_\ell(\text{RMSNorm}(\tilde{z}_\ell)), \end{aligned} \quad (4)$$

Model	Typed	BANKING77		MASSIVE		Mind2Web		
	Test	$K = 8$	$K = 16$	$K = 8$	$K = 16$	Task	Website	Domain
■ Jev-1.13.0	74.10	86.14	82.50	84.63	81.61	66.26	59.97	62.47
■ JevK5	62.30	78.18	69.03	80.16	69.70	51.47	43.30	45.60
■ InternDecision-4B	78.70	80.78	75.03	81.20	75.79	48.02	41.17	43.61
■ InternDecision-2B	75.85	48.31	35.10	45.33	41.69	29.43	27.47	26.12
■ InternDecision-0.8B	69.35	29.74	22.63	28.58	24.61	25.33	22.53	21.48
■ Re-Flex-4B	58.85	78.77	74.38	76.63	71.39	52.08	45.05	47.64
■ SemiF-4B	62.75	69.97	67.18	69.70	67.38	48.60 [†]	41.16 [†]	43.67 [†]
■ KEV-4B (BF16)	66.00	83.47	82.47	76.60	73.54	40.32	34.17	35.62
■ KEV-4B (FP32)	66.05	83.44	82.34	76.53	73.64	40.53	34.25	35.81
■ CLM-8B	43.05	26.95	18.15	33.36	23.23	21.73	21.16	19.79
■ Valen-Preview-2B	37.75	19.61	19.09	19.60	21.72	13.27	14.16	14.38
■ Laya-English	34.50	58.05	50.49	44.69	37.63	15.96	11.34	15.26
■ Qwen3.5-4B label LP	50.85	49.84	43.12	47.31	41.63	18.14	15.30	18.34
■ Qwen3.5-2B label LP	41.70	48.25	35.65	30.93	25.25	12.46	10.73	11.47
■ Jeva-4B	78.30	92.05	91.62	91.09	90.22	79.99	62.25	66.23

Table 1: **First-argmax accuracy (% , higher is better), including null outcomes where applicable.** Mind2Web uses $K = 16$. [†]Supported-only accuracy; coverage is 99.2%, 99.8%, and 97.3% respectively; all other entries have full coverage. Dark-to-light gold marks the top three distinct full-coverage scores per column.

where F_ℓ is a bias-free SwiGLU network and $\alpha_\ell = \tau_\ell \sigma(\gamma_\ell) / \sqrt{L}$. This coupling ensures $0 < \alpha_\ell / \tau_\ell < 1 / \sqrt{L}$. All temperatures are learned positive scalars. Only z_ℓ evolves; keys and values depend solely on the unchanged options.

Proposition 1 (Set-dependent preference with one block). *There exist fixed Jeva parameters and two candidate sets containing the same options A, B such that, for the same request, a single block uniquely selects A in one set and B in the other. No independent scorer in Eq. (2) with set-independent tie-breaking can realize both choices.*

Proof. Use one two-dimensional head, identity projections, and zero FFN output. Score each candidate by $s_i = z_L^\top u_i$. Let $z_0 = (1, 0)$, $A = (4, 3)/5$, $B = (4, -3)/5$, and $C_\pm = (3, \pm 4)/5$. Compare sets (A, B, C_+) and (A, B, C_-) . All vectors are unit length. With $\tau = 3/20$ and $\gamma = -\log 2$, the residual scale is $\alpha = 1/20$; the third option receives weight $w = (1 + 2e^{4/3})^{-1}$. The score differences are

$$s_A(\mathcal{O}_\pm) - s_B(\mathcal{O}_\pm) = \pm 24\alpha w / 25. \quad (5)$$

The winning A or B also exceeds $C_\pm$: its initial advantage $1/5$ can shrink by at most $2\alpha < 1/5$. The winners are therefore unique, whereas an independent scorer cannot reverse their ordering by Eq. (2). $\square$

Figure 3 shows this construction; Appendix B.1 embeds it in the deployed width and head count.

Thus reusable features do not imply independent decisions. The mixer forms the message directly, without storing a $K \times K$ interaction matrix.

3.3 Competition, Coverage, and Typed Outputs

Jeva computes final scores $s_i = z_L^\top u_i$ and conditional candidate probabilities $q = \text{softmax}(s / \tau_{\text{dec}})$. Neither the initial nor the final state is unit-normalized, retaining magnitude information for confidence estimation. A common score shift leaves q unchanged: relative preference alone cannot establish whether any candidate is acceptable. To form a full distribution p over candidates and null ($\perp$), Jeva separately estimates the probability ρ that at least one acceptable candidate is present:

$$\rho = \sigma\left(\frac{\max_i s_i - b_{\text{cov}}}{\tau_{\text{cov}}}\right), \quad (6)$$

$$p_\perp = 1 - \rho, \quad p_i = \rho q_i.$$

Here, b_{cov} is a learned coverage threshold and $\tau_{\text{cov}} > 0$ is a learned temperature. Writing Y for the decision outcome, this implements $P(Y = i) = P(Y \neq \perp)P(Y = i \mid Y \neq \perp)$. Closed-set requests use q ; open-set requests use p .

Proposition 2 (Coverage without a cardinality denominator). *At fixed final scores, adding options below the existing maximum leaves ρ unchanged. In contrast, a flat softmax with null score b_0 and K identical candidate scores s assigns*

$$p_\perp^{\text{flat}} = \sigma((b_0 - s) / \tau - \log K). \quad (7)$$

Model	Noul ↓		Score ↓		Choice permutations		
	Brier	MAE	MAE	RMSE	Flip (%) ↓	All-order (%) ↑	Mean TV ↓
■ Jev-1.13.0	0.0342	0.1456	0.3744	0.4795	17.00	66.67	0.0858
■ JevK5	0.0987	0.2448	0.4370	0.5935	26.17	47.83	0.116
■ InternDecision-4B	0.0340	0.1431	0.2488	0.3208	22.67	64.50	0.106
■ InternDecision-2B	0.0440	0.1649	0.2641	0.3433	26.83	60.33	0.124
■ InternDecision-0.8B	0.0503	0.1835	0.3145	0.4013	45.00	45.33	0.135
■ Re-Flex-4B	0.1768	0.3464	0.4795	0.6490	26.50	54.67	0.116
■ SemiF-4B	0.0913	0.2439	0.4722	0.6134	32.33	50.33	0.165
■ KEV-4B (BF16)	0.0888	0.2282	0.3987	0.5669	17.50	55.50	0.0658
■ KEV-4B (FP32)	0.0887	0.2283	0.3989	0.5675	17.17	55.83	0.0657
■ Valen-Preview-2B	0.1509	0.3348	0.7786	0.9274	31.17	24.17	0.139
■ Laya-English	0.1514	0.3206	0.7083	0.8383	66.83	11.83	0.139
■ Qwen3.5-4B label LP	0.1623	0.3213	0.8095	1.0235	84.50	14.17	0.414
■ Qwen3.5-2B label LP	0.1411	0.3224	0.6221	0.7730	76.83	18.17	0.199
■ Jeva-4B	0.0184	0.0939	0.2315	0.3222	0.00	76.67	3.17e-08

Table 2: **Typped fidelity and order stability.** Flip measures any decision change; all-order accuracy requires correctness under every tested order. Mean TV compares candidate-aligned probability distributions.

Proof. Adding scores below the existing maximum leaves ρ unchanged. For flat softmax, dividing numerator and denominator by $e^{b_0/\tau}$ gives $p_{\perp}^{\text{flat}} = 1/(1 + K e^{(s-b_0)/\tau})$, equivalent to Eq. (7). $\square$

Figure 4 illustrates these effects through controlled changes to the final scores. Set refinement may still change scores, and the joint argmax over p need not remain fixed when candidates are added (Appendix A.3).

The same computation supports CHOICE through candidate selection, NOUL through the positive Boolean candidate’s probability q_{i+} , and SCORE through $\sum_i q_i v_i$ for numeric rubric values v_i . Rubric values and the positive designation determine output interpretation, not extra encoder inputs. Separate temperatures govern evidence aggregation, competition, and coverage.

3.4 Symmetry, Reuse, and Scaling

An atomic option memory stores both its semantic vector and projected views:

$$\mathcal{M}_t(o_i) = \left(u_i, \{k_{\ell_i}^h, v_{\ell_i}^h\}_{\ell,h} \right). \quad (8)$$

Requests gather these entries in their current candidate order. Our set cache stores assembled candidate tuples; the hybrid cache additionally reuses individual options across newly assembled sets. State encoding and decision refinement still execute for every request. Figure 5 illustrates this distinction.

Proposition 3 (Candidate symmetry and exact reuse). *With fixed weights and deterministic exact-arithmetic evaluation, permuting candidates and*

their metadata permutes candidate probabilities but leaves z_L , null probability, and typed Noul/Score outputs unchanged. Replacing option computations by identical cached entries also leaves all outputs unchanged.

Proof. Reordering keys and values only reindexes the sum in Eq. (3). Induction through the state updates therefore preserves z_L ; shared scoring permutes its outputs. Cached entries are identical because Eq. (8) contains no request or other candidate. Substitution preserves every subsequent operation. $\square$

A unique winning option is invariant as a semantic item. Ties and finite-precision effects are discussed in Appendix B. No order-consistency loss is required. At fixed width d and FFN width f , prepared-memory decision cost is $O(L(d^2 + Kd + df))$, linear in K . A bank of M options stores $O(M(L + 1)d)$ values. New options require encoding and projection, but neither additional output parameters nor a joint token window. Each text has its own encoder length budget; this does not remove individual-text length limits. The $\Omega(K)$ cost of emitting a dense distribution already matches the head’s order in candidate count.

3.5 Direct Supervised Learning

For normalized open-set targets $(y_{\perp}, y_1, \dots, y_K)$, let $c_y = \sum_i y_i$ and $g = \text{logit}(\rho)$. We minimize

$$\ell = \text{BCEWithLogits}(g, c_y) - \sum_i y_i \log q_i. \quad (9)$$

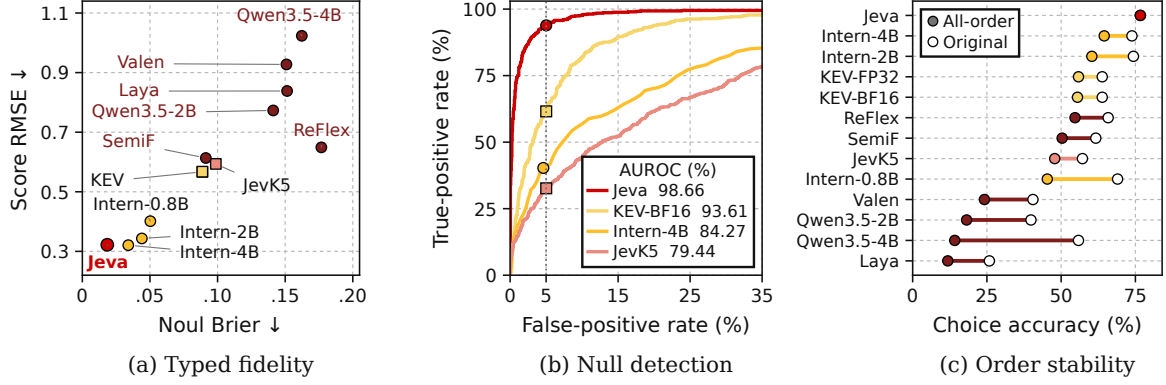


Figure 6: Decision quality and stability. (a) Noul Brier versus Score RMSE; lower is better, with different units. (b) Banking77 $K = 16$ null ROC: the legend gives full AUROC; markers show each measured near-5% FPR point. (c) Original and all-order accuracy on 600 Choice examples; filled markers denote all-order accuracy, and segments show permutation sensitivity.

This primary loss is joint cross-entropy on p : all-null examples supervise availability without imposing a spurious candidate ranking. Closed sets use candidate cross-entropy. Stage-dependent Noul and Score auxiliaries, initialization, and optimization details appear in Appendix B.

Jeva-4B uses Qwen3.5-4B with pooled width 2560, decision width $d = 768$, $L = 1$, $H = 8$, and $f = 3072$. We train rank-64 LoRA (Hu et al., 2022) with both projections and the decision head (13.37M parameters), using separate AdamW parameter groups and global gradient clipping. The curriculum spans typed decisions, intent selection, verification, reading/reasoning, and large-candidate retrieval; stage transitions retain the architecture and preceding weights. LoRA is merged for serving, and memories are prepared for the resulting fixed model.

4 Experiments

We evaluate decision quality, structural properties, and efficiency using a common suite and controlled permutation, batching, and cache tests. The suite includes Typed Decisions (LocalLLaMA, 2026) (2,000 test decisions), BANKING77 (Casanueva et al., 2020) and MASSIVE (FitzGerald et al., 2023) with 8 or 16 candidates, and three Mind2Web (Deng et al., 2023) slices selecting from 16 supplied candidates. Baselines retain their documented interfaces and native runtimes. For open-set evaluation, JevK5 and CLM receive an explicit None candidate, while typed baselines such as Jev, InternDecision, and KEV answer an auxiliary Noul question about candidate availability. For the latter, estimated availability scales the conditional candidate probabilities, with

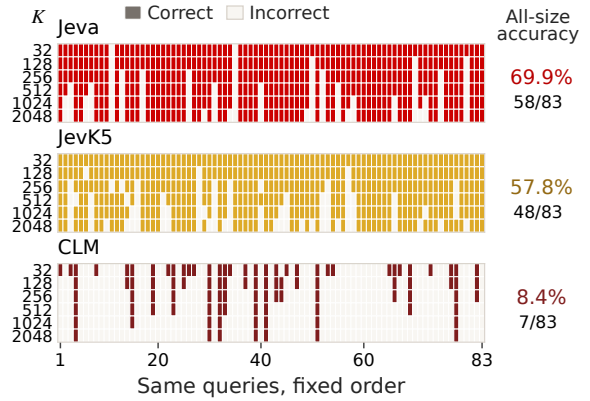


Figure 7: Correctness across nested candidate sets. Columns show 83 fixed queries; rows indicate candidate count K . Colored cells denote correct predictions. All-size accuracy requires correctness at every tested size.

the remaining mass assigned to null. Accuracy uses first-argmax predictions, including null where applicable, on fully supported slices; unsupported requests are reported separately. Local latency uses synchronized H200 timing; within-model comparisons fix the checkpoint and query stream.

4.1 General and typed decision quality

Table 1 shows that Jeva leads seven of eight slices in full-coverage accuracy, exceeding 90% on BANKING77 and MASSIVE at both candidate counts. It achieves 79.99%, 62.25%, and 66.23% on Mind2Web cross-task, cross-website, and cross-domain selection, respectively. On Typed Decisions, Jeva reaches 78.30%, compared with InternDecision-4B’s 78.70%.

Table 2 evaluates 600 Noul and 800 Score examples. Jeva achieves the lowest Noul Brier (0.0184) and MAE (0.0939) against soft probability targets, and the lowest Score MAE (0.2315). Its Score RMSE is 0.3222, close to InternDecision-4B’s

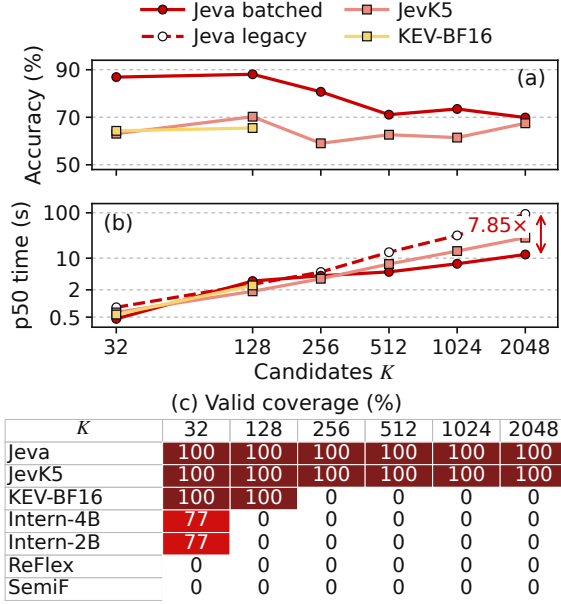


Figure 8: **ToolRet across six disjoint candidate-count cohorts.** (a) Full-support tied-top accuracy. (b) Natural-stream p50 latency; Jeva uses 256-row/131,072-token batching, with legacy encoding dashed. (c) Valid coverage; 77% denotes 65/84 requests. Unsupported slices are excluded from accuracy.

0.3208. Score errors compare predicted expectations with targets in the original rubric units.

We evaluate 600 Choice examples in the original, reversed, and up to three seeded random orders, aligning candidate identities. Jeva records zero flips, mean TV of 3.17×10^{-8} , and identical original and all-order accuracy of 76.67%, consistent with Proposition 3. Jev-1.13.0, InternDecision-4B, and JevK5 exhibit flip rates of 17.00%, 22.67%, and 26.17%, with all-order accuracies of 66.67%, 64.50%, and 47.83%, respectively. All-order accuracy requires correctness under every tested order.

4.2 Null detection

On 3,080 matched BANKING77 requests with 16 candidates (588 answer-absent), Jeva achieves 98.66% null-detection AUROC, versus 93.61% for KEV-BF16, 84.27% for InternDecision-4B, and 79.44% for JevK5. It detects 93.88% of absent answers at 5.02% FPR (Figure 6(b)). Baselines use auxiliary coverage Noul outputs or an explicit None candidate. Near-5% FPR thresholds are each model’s empirical 95th percentile of answer-present scores; strictly higher scores predict null.

4.3 Candidate scaling and batching

Figure 8(a,c) reports accuracy and coverage on 500 ToolRet requests in distinct-query cohorts with

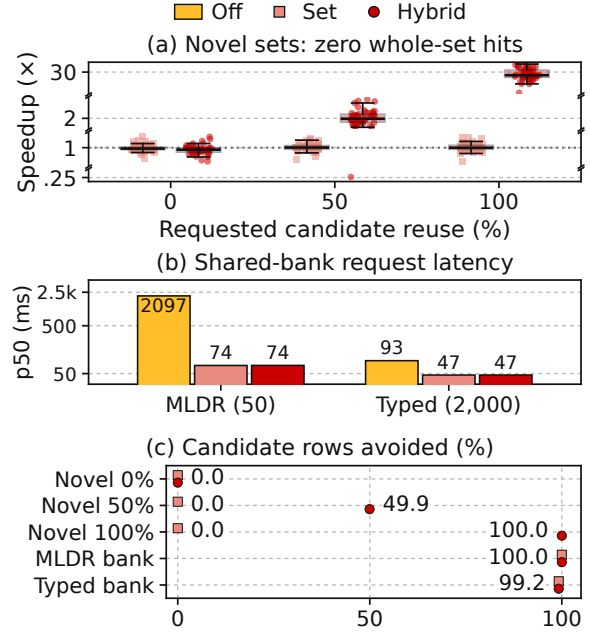


Figure 9: **Candidate reuse** (32-row/16,384-token batching). (a) Paired speedups over Cache-Off: 50 novel-set requests per reuse level, zero set hits, broken axis; boxes show median/IQR with 1.5-IQR whiskers. (b) Shared-bank p50 latency. (c) Encoded rows avoided. Priming is timed separately; no answers are cached.

$K \in \{32, 128, 256, 512, 1024, 2048\}$. Positives are retained (389 multi-positive requests), with weak or unjudged added negatives. Any highest-target-score candidate counts as correct.

Jeva covers all requests with 78.40% aggregate accuracy, versus 64.00% for recursive JevK5, while scoring each complete set in one decision-head pass. Both InternDecision variants support only 65/84 requests at $K = 32$; larger cohorts exceed their registered limits. Unsupported slices are excluded from accuracy comparisons.

Figure 8(b) shows the effect of length-aware batching (256 rows/131,072 padded tokens), which preserves cohort accuracies, candidate order, and cache policy. Compared with chunk-2 encoding, p50 latency falls from 31.730 to 7.519 s at $K = 1024$ and from 94.459 to 12.040 s at $K = 2048$ (7.85 $\times$). At $K = 2048$, Jeva achieves 69.88% accuracy, versus JevK5’s 67.47% at 28 s. Encoder batching retains full-set access in the decision head.

Robustness to candidate-set expansion. Figure 7 evaluates the same 83 queries as nested candidate sets expand from 32 to 2,048 options. Jeva remains correct at every size on 58 queries, versus 48 for JevK5 and 7 for CLM. Its all-size accuracy is 69.88%, exceeding JevK5 by 12.05 pts.

4.4 Reuse across states and candidate sets

Figure 9 compares Cache-Off, exact-set caching (Set), and additional individual-option reuse (Hybrid), using fixed weights, cyclic mode ordering, and cache resets. Only candidate computations are cached, never previous answers.

For 50 matched MLDR queries at 0%, approximately 50%, and 100% option reuse, every novel-set lookup misses the whole-set cache. Hybrid achieves $1.95\times$ and $27.98\times$ speedups in p50 latency at partial and full reuse, while Set remains near 2.1 s. At full reuse, p50 falls from 2,102.0 to 75.1 ms; the median paired speedup is $27.86\times$.

For shared banks, Typed Decisions p50 falls from 93.1 ms to 46.6 ms (Set) and 46.7 ms (Hybrid), avoiding 99.2% of candidate-row encodings. Primed MLDR p50 falls from 2,097.3 ms to 73.9/74.2 ms. Typed timings include cold starts; MLDR priming is timed separately and excluded. No decisions change across conditions; the maximum absolute probability difference is 0.00736.

5 Conclusion

We introduce Jeva, a native architecture specifically designed for system one decision makings. Jeva models decisions as an evolving state reading reusable option memory, connecting permutation symmetry, variable candidate counts, explicit null probability, and caching. The trained Jeva-4B combines broad decision capability with strong null detection, stable predictions under reordering, and savings from hybrid cache. Large-set results show the practical value of candidate-indexed readout.

References

- Diogo Almeida. 2026. Introducing System One Models & Jev. <https://typesafe.ai/blog/introducing-system-one-models-and-jev>. TypeSafe AI Blog.
- Iñigo Casanueva, Tadas Temčinas, Daniela Gerz, Matthew Henderson, and Ivan Vulić. 2020. [Efficient intent detection with dual sentence encoders](#). In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 38–45.
- Jianlyu Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. [M3-Embedding: Multi-linguality, multi-functionality, multi-granularity text embeddings through self-knowledge distillation](#). In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2318–2335.
- Convai Innovations. 2026. Laya: Multilingual, non-autoregressive System 1 decision engine. <https://github.com/NandhaKishorM/laya>. Code and model release.
- Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. 2023. [Mind2Web: Towards a generalist agent for the web](#). In *Advances in Neural Information Processing Systems*, volume 36.
- Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natara-jan. 2023. [MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302. Association for Computational Linguistics.
- Yonatan Geifman and Ran El-Yaniv. 2017. [Selective classification for deep neural networks](#). In *Advances in Neural Information Processing Systems*, volume 30.
- Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. 2017. [On calibration of modern neural networks](#). In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1321–1330.
- Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. [LoRA: Low-rank adaptation of large language models](#). In *International Conference on Learning Representations*.
- InternLM. 2026. Intern-Decision: Fast multi-modal decision model. <https://github.com/InternLM/Intern-Decision>. Code and model release.
- Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. 2021. [Perceiver: General perception with iterative attention](#). In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4651–4664.
- JevK5 authors. 2026. JevK5: An open-weight Jev alternative. <https://github.com/allebee/jevk5>. 4B model v0.3.
- Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. [Dense passage retrieval for open-domain question answering](#). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781.

- Omar Khattab and Matei Zaharia. 2020. [ColBERT: Efficient and effective passage search via contextualized late interaction over BERT](#). In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*.
- Jacky Kwok, Hango Kang, Tarun Suresh, Jon Saad-Falcon, Marco Pavone, Christopher Ré, and Azalia Mirhoseini. 2026. Contrastive language models. <https://github.com/Contrastive-LM/CLM>. Technical blog and official implementation.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiosek, Seungjin Choi, and Yee Whye Teh. 2019. [Set transformer: A framework for attention-based permutation-invariant neural networks](#). In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3744–3753.
- Theodore Lee. 2026. SemIf: Semantic ifs from open models. <https://github.com/TheoLeeCJ/SemIf-OpenJev>. GitHub repository; formerly OpenJev.
- LocalLLaMA. 2026. Typed decisions. <https://huggingface.co/datasets/LocalLLaMA/typed-decisions>. Hugging Face dataset.
- Jared Palmer. 2026. Kev-4B: A calibrated decision model on Qwen3.5-4B. <https://huggingface.co/jaredpalmer/kev-4b>. Kev 1.0.
- Zhen Qin, Rolf Jagerman, Kai Hui, Honglei Zhuang, Junru Wu, Le Yan, Jiaming Shen, Tianqi Liu, Jialu Liu, Donald Metzler, Xuanhui Wang, and Michael Bendersky. 2024. [Large language models are effective text rankers with pairwise ranking prompting](#). In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1504–1518.
- Kshetrajna Raghavan. 2026. reflex: A small open decision model. <https://github.com/kshetrajna12/reflex>. Code and model release.
- Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. [Know what you don’t know: Unanswerable questions for SQuAD](#). In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.
- Marzieh Saeidi, Max Bartolo, Patrick Lewis, Sameer Singh, Tim Rocktäschel, Mike Sheldon, Guillaume Bouchard, and Sebastian Riedel. 2018. [Interpretation of natural language rules in conversational machine reading](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2087–2097.
- Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551.
- Zhengliang Shi, Yuhan Wang, Lingyong Yan, Pengjie Ren, Shuaiqiang Wang, Dawei Yin, and Zhaochun Ren. 2025. [Retrieval models aren’t tool-savvy: Benchmarking tool retrieval for large language models](#). In *Findings of the Association for Computational Linguistics: ACL 2025*, pages 24497–24524.
- Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. 2017. [Deep sets](#). In *Advances in Neural Information Processing Systems*, volume 30.
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. [Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623.

A Derivations and Structural Properties

A.1 Entropy-Regularized Candidate Reading

For compatibility vector c and probability simplex Δ_K , the attention weights solve

$$a_\ell^h = \arg \max_{a \in \Delta_K} \left\{ a^\top c + \tau_\ell \mathcal{H}(a) \right\}, \quad (10)$$

$$\mathcal{H}(a) = - \sum_i a_i \log a_i.$$

This entropy-regularized selection balances compatibility against the spread of evidence. Learned value and output matrices determine how that evidence changes the state.

Why this aggregation rule? Its variational characterization is exact. Let $Z = \sum_i \exp(c_i/\tau_\ell)$ and $a_i^* = \exp(c_i/\tau_\ell)/Z$. For every $a \in \Delta_K$, using $0 \log 0 = 0$,

$$a^\top c + \tau_\ell \mathcal{H}(a) = \tau_\ell \log Z - \tau_\ell D_{\text{KL}}(a \| a^*). \quad (11)$$

Since $D_{\text{KL}}(a \| a^*) \geq 0$, the objective is at most $\tau_\ell \log Z$, attained uniquely at $a = a^*$. This proves Eq. (10). The weights are therefore the unique entropy-regularized allocation of attention over candidate evidence. They are intermediate read weights, not the final decision probabilities: the state may gather evidence from several options before committing to one.

Block parameterization. Here $F_\ell(r) = W_{\ell, \text{down}}[\text{SiLU}(W_{\ell, \text{gate}} r) \odot W_{\ell, \text{up}} r]$ is a bias-free SwiGLU network with hidden width f . We initialize the four decision projection matrices to identity and the SwiGLU output matrix to zero. Subsequent optimization learns both branches. Each temperature is parameterized as $\tau = 10^{-4} + \text{softplus}(\eta)$ with its own trainable scalar η . The coupling gives $0 < \alpha_\ell/\tau_\ell < 1/\sqrt{L}$, scaling the mixing coefficient with both selection temperature and decision depth.

Separate temperatures for separate questions.

τ_ℓ controls evidence aggregation within a block, τ_{dec} controls the final competition, and τ_{cov} controls the slope of the answer-availability gate. For fixed logits c and $a(\tau) = \text{softmax}(c/\tau)$, $d\mathcal{H}(a(\tau))/d\tau = \text{Var}_{i \sim a(\tau)}(c_i)/\tau^3 \geq 0$ (Appendix B.2). Decoupling the parameters therefore lets training tune evidence mixing without forcing an identical readout temperature. The numerical temperatures need not be directly comparable: per-head cosine similarities and final state–option dot products have different scales.

A.2 Geometry and Single-Block Expressivity

Geometry of the selection correction. The candidate interaction can be seen directly in score space. Let S_h insert a head vector into the concatenation, let $W_\ell^{V,h}$ be the corresponding value-projection rows, and define

$$G_{\ell, ij}^h = u_i^\top W_\ell^O S_h W_\ell^{V,h} u_j. \quad (12)$$

Before the channel-refinement branch, the score of candidate i changes by

$$u_i^\top \tilde{z}_\ell - u_i^\top z_\ell = \alpha_\ell \sum_{h,j} a_{\ell j}^h G_{\ell, ij}^h. \quad (13)$$

The distinction from independent scoring already appears with one block.

We give the construction for Proposition 1.

Proof. In decision representation space, use a two-dimensional, single-head block with identity Q/K/V/O projections and zero FFN output. Let

$$\begin{aligned} z_0 &= (1, 0), & A &= (4, 3)/5, \\ B &= (4, -3)/5, & C_\pm &= (3, \pm 4)/5, \\ \mathcal{O}_\pm &= (A, B, C_\pm). \end{aligned} \quad (14)$$

All vectors are unit length, so query/key normalization leaves them unchanged. Choose $\tau_{\text{set}} = 3/20$ and $\gamma = -\log 2$, giving $\alpha = 1/20$. The weight of $C_\pm$ is $t = (1 + 2e^{4/3})^{-1}$. The two refined states are $z_\pm = (1 + \alpha(4 - t)/5, \pm 4\alpha t/5)$, hence

$$s_A(\mathcal{O}_\pm) - s_B(\mathcal{O}_\pm) = \pm \frac{24\alpha t}{25}. \quad (15)$$

The winning A or B also exceeds $C_\pm$: its initial score advantage is $1/5$, and a correction of norm at most α can reduce a pairwise margin by at most $2\alpha < 1/5$. Thus both winners are unique. An independent scorer cannot reverse this relative ordering by Eq. (2); equal scores with a fixed tie-breaking rule also select the same one of A, B in both sets. To embed the construction into H heads of width at least two, place one copy of each two-dimensional vector in every head, zero-pad the remaining coordinates, and scale each copy by $H^{-1/2}$. Per-head query/key normalization recovers the original attention weights, while the summed headwise dot products reproduce the original scores. $\square$

Thus, evidence selected from one candidate can affect another candidate’s score through learned geometry, without updating either candidate. For a single head with identity value/output maps, this

reduces to the Gram correction $\alpha_\ell U U^\top a_\ell$. The deployed multi-head model learns a more general interaction. It computes the weighted message first, never materializing a $K \times K$ matrix.

The identity in Eq. (13) explains how fixed candidate vectors still support relational decisions. Candidate j contributes to candidate i through $G_{\ell,ij}^h$, weighted by its relevance to the current state. The matrix is an algebraic description of the interaction, not a stored tensor.

The proposition identifies a representational capability, not a guarantee that every set change should reverse a preference. It shows why one decision block is already functionally distinct from an independent similarity readout. Its state update can reflect relations within the current set while retaining cacheable candidate features. Figure 3 visualizes the construction. Conversely, low total attention on added alternatives gives an explicit bound on their influence; Appendix B.3 derives this bound for the actual matrix-valued mixer.

A.3 Coverage and Candidate Count

The following expands Proposition 2. Fix the final candidate scores. Adding scores no larger than the existing maximum leaves ρ in Eq. (6) unchanged. By contrast, a flat softmax with a fixed null score b_0 and K equal candidate scores s gives

$$\begin{aligned} p_{\perp}^{\text{flat}} &= \frac{1}{1 + K \exp((s - b_0)/\tau)} \\ &= \sigma\left(\frac{b_0 - s}{\tau} - \log K\right). \end{aligned} \quad (16)$$

Proof. The maximum, and thus the argument of the coverage sigmoid, is unchanged under the stated addition. For flat softmax, summing the K identical candidate exponentials and dividing by the null exponential gives Eq. (16). $\square$

Proposition 2 removes a compulsory $\log K$ contribution to the *coverage* decision. Set refinement can still change the scores, and competition still redistributes probability across options. In particular, a joint argmax over null and individual candidates is distinct from thresholding coverage. The factorization assigns these effects to explicit components that can be trained and examined separately. Figure 4 illustrates the two components with fixed-score examples.

A.4 Detailed Reuse and Symmetry Proofs

Exact reuse. The following expands the reuse part of Proposition 3.

Proof. Equation (8) contains no state, question, or other candidate. A stored entry is thus the same value as a recomputed entry. Gathering entries in the requested order supplies identical inputs to every block. Induction through Eq. (4) yields identical states, followed by identical competition, coverage, and typed readouts. $\square$

The result concerns reuse of *computation*, not reuse of an answer. Figure 5 illustrates reuse across new requests whose candidate sets overlap without repeating. Even with every option cached, the request is encoded and the full decision module is evaluated. Both semantic vectors and layer-specific key/value projections can be reused: learned matrix views do not break cacheability when they depend only on an unchanged option.

With FFN width f , preparing layer views costs $O(LKd^2)$, while a request with prepared memories costs $O(L(d^2 + Kd + df))$ for the decision module, in addition to state encoding. For $f = 4d$, this is $O(L(d^2 + Kd))$. A bank of M entries stores $O(M(L + 1)d)$ numbers. Total computation is the sum of text encoding and decision costs: encoding the request and any uncached options, preparing missing views, and applying the linear-in- K candidate interaction. Reuse removes the corresponding option-side terms. Producing a dense distribution over all K candidates itself requires $\Omega(K)$ output writes. At fixed width and depth, the prepared-memory head therefore has optimal order of growth in K for this output interface. This is a computation bound, not an accuracy extrapolation claim. Independent encoding also assigns a separate token budget to each text, rather than requiring the entire option list to fit one concatenated context window.

Candidate symmetry. We expand the symmetry part of Proposition 3.

Proof. Let π be a permutation of $\{1, \dots, K\}$, with candidates and their metadata reordered together, so $u'_i = u_{\pi(i)}$. Independent encoding leaves $z'_0 = z_0$ and permutes the option rows. If $z'_\ell = z_\ell$, the queries agree, keys and values permute, and the softmax denominator is unchanged. Therefore $a'^h_{\ell i} = a^h_{\ell, \pi(i)}$, and reindexing gives

$$\sum_i a'^h_{\ell i} v'^h_{\ell i} = \sum_j a^h_{\ell j} v^h_{\ell j}. \quad (17)$$

Both residual branches receive the same state and message, so $z'_{\ell+1} = z_{\ell+1}$. Induction yields $z'_L = z_L$. Final dot products and their softmax permute,

whereas their maximum and coverage do not. Score values move with their candidates, so reindexing preserves the weighted expectation. Noul continues to read the same positive candidate’s probability. A unique maximizing candidate retains its identity. $\square$

This structural symmetry requires no permutation augmentation or consistency loss. It concerns reorderings of the same alternatives; candidate multiplicity and set composition remain meaningful inputs. Numerical tolerances and the tied-maximum case are described in Appendix B.

A.5 Joint Cross-Entropy Identity

The open-set loss in Eq. (9) is joint cross-entropy. Indeed,

$$\begin{aligned} & -y_{\perp} \log p_{\perp} - \sum_i y_i \log p_i \\ &= -y_{\perp} \log(1 - \rho) - c_y \log \rho \\ & \quad - \sum_i y_i \log q_i \\ &= \text{BCEWithLogits}(g, c_y) \\ & \quad - \sum_i y_i \log q_i, \end{aligned} \quad (18)$$

using $y_{\perp} + c_y = 1$ and $\rho = \sigma(g)$. Under this primary joint log loss, all-null examples supervise coverage without assigning a relative-ranking target; answer-present examples supervise both coverage and competition. The corresponding population log-loss decomposition is given in Appendix B.4.

B Method Proofs and Implementation Details

B.1 The Single-Block Construction in Higher Dimensions

Proposition 1 is an existence statement in decision representation space. To embed its two-dimensional vectors in H heads of width $d_h \geq 2$, define

$$J(v) = H^{-1/2} \text{Concat}_{h=1}^H(v_1, v_2, 0, \dots, 0).$$

Then $J(v)^{\top} J(w) = v^{\top} w$. For unit v , each nonzero head slice has norm $H^{-1/2}$; normalizing queries and keys cancels this scale. Every head therefore uses the two-dimensional attention weights. Values retain the $H^{-1/2}$ factor, and identity output projection gives $z_1 = J(z_{\pm})$ after mixing. Its readout scores are exactly those

of the original construction. This embeds it in $d = 768, H = 8$. Setting $W_{\text{down}} = 0$ gives zero FFN output independently of RMSNorm, completing the construction within the deployed architecture.

For completeness, if an independent scorer ties A and B , a fixed tie-breaking rule orders them identically in both sets. Equal scores therefore cannot resolve the contradiction in the proof. A common coverage factor also cancels from p_A/p_B , so adding a coverage gate to an otherwise independent scorer does not remove this restriction.

B.2 Temperature and the Sensitivity of Candidate Aggregation

Entropy at fixed compatibility. For fixed finite logits c_i , write $a_i(\tau) = e^{c_i/\tau}/Z(\tau)$ and $\mu(\tau) = \sum_i a_i(\tau) c_i$. Differentiating gives

$$\frac{da_i}{d\tau} = \frac{a_i(\mu - c_i)}{\tau^2}, \quad \frac{d\mu}{d\tau} = -\frac{\text{Var}_{a(\tau)}(c)}{\tau^2}.$$

Since $\mathcal{H}(a) = \log Z - \mu/\tau$ and $d \log Z/d\tau = -\mu/\tau^2$, the two mean terms cancel, yielding

$$\frac{d\mathcal{H}(a(\tau))}{d\tau} = \frac{\text{Var}_{a(\tau)}(c)}{\tau^3} \geq 0. \quad (19)$$

For fixed logits, attention entropy is therefore non-decreasing in temperature.

Why the matrix model involves cross-covariance.

Fix one head’s candidate keys and values, and treat its normalized query q as the input to $\mu_v(q) = \sum_i a_i(q) v_i$. Put $\bar{k} = \sum_i a_i k_i$. The softmax derivative gives $\nabla_q a_i = a_i(k_i - \bar{k})/\tau$, hence

$$\begin{aligned} D_q \mu_v &= \frac{1}{\tau} \left[\sum_i a_i v_i k_i^{\top} - \mu_v \bar{k}^{\top} \right] \\ &= \frac{\text{Cov}_a(v, k)}{\tau}. \end{aligned} \quad (20)$$

If $\|k_i\| \leq 1$ and $\|v_i\| \leq R_v$, this derivative has operator norm at most R_v/τ . Indeed, for unit vectors r, w , Cauchy–Schwarz bounds $|\text{Cov}(r^{\top} v, w^{\top} k)|$ by the square root of the two variances, which are at most R_v^2 and 1.

The learned matrix model uses different keys and values, so this cross-covariance need not be symmetric or positive semidefinite. To differentiate a complete block with respect to z , one must also apply the query projection and normalization derivatives, the output projection, and the RMSNorm–SwiGLU residual. The coupling $\alpha/\tau < 1/\sqrt{L}$ bounds the scalar prefactor of the cross-covariance contribution within this derivative composition.

B.3 A Quantitative Bound for Added Candidates

Set dependence is useful when alternatives supply relevant evidence. For weakly attended additions, its magnitude can be bounded. Fix one head’s incoming query and the original keys/values. Add r candidates whose compatibility logits are at most $c_{\max} - \delta$, where $c_{\max}$ is the original maximum and $\delta \geq 0$. Let β be their total attention mass after insertion. Then

$$\beta \leq \frac{re^{-\delta/\tau}}{1 + re^{-\delta/\tau}}, \quad (21)$$

$$\|\mu_{\text{new}} - \mu_{\text{old}}\| \leq 2R_v\beta$$

provided every old and new value has norm at most R_v .

Proof. Let $Z = \sum_{\text{old}} e^{c_i/\tau}$ and $R = \sum_{\text{new}} e^{c_i/\tau}$. Since $Z \geq e^{c_{\max}/\tau}$ and $R \leq re^{(c_{\max}-\delta)/\tau}$, $\beta = R/(Z + R)$ obeys the first bound. The original weights are all rescaled by $1 - \beta$. Writing μ_{add} for the normalized mean over new values gives the exact identity $\mu_{\text{new}} = (1 - \beta)\mu_{\text{old}} + \beta\mu_{\text{add}}$. Both means lie in a radius- R_v ball, which proves the second bound. $\square$

For the actual multi-head message, let the corresponding masses and value-norm bounds be β_h, R_h . Concatenation and the output matrix give

$$\|\Delta m\| \leq 2\|W^O\|_2 \left(\sum_h R_h^2 \beta_h^2 \right)^{1/2}. \quad (22)$$

In the single-block model, the initial state is independent of the set, so its incoming query is indeed fixed across this comparison. If $F \circ \text{RMSNorm}$ has Lipschitz constant B on the segment between the two post-mixer states, then

$$\|\Delta z_1\| \leq (1 + B)\alpha\|\Delta m\|,$$

$$|\Delta \log(q_i/q_j)| \leq \frac{2\|\Delta z_1\|}{\tau_{\text{dec}}}$$

for any two unchanged options. The first inequality follows by the triangle inequality through the residual FFN, and the second uses $\|u_i - u_j\| \leq 2$. The bound depends on the additions’ total attention mass and the local FFN sensitivity. Extending it across multiple blocks additionally requires bounds on changes in the incoming state.

B.4 Joint Log Loss and the Meaning of Null

Let the true conditional output distribution be $p^* = (1 - \rho^*, \rho^*q^*)$ and the model distribution be $p = (1 - \rho, \rho q)$. Expanding the KL divergence gives

$$\begin{aligned} D_{\text{KL}}(p^* \| p) &= (1 - \rho^*) \log \frac{1 - \rho^*}{1 - \rho} + \rho^* \log \frac{\rho^*}{\rho} \\ &\quad + \rho^* \sum_i q_i^* \log \frac{q_i^*}{q_i} \\ &= D_{\text{KL}}(\text{Ber}(\rho^*) \| \text{Ber}(\rho)) \\ &\quad + \rho^* D_{\text{KL}}(q^* \| q). \end{aligned} \quad (23)$$

The excess expected primary joint log loss decomposes into coverage error and conditional competition error, with the latter weighted by ρ^* . If $\rho^* = 0$, the competition term vanishes and q^* can be chosen arbitrarily.

Coverage and the joint maximum-probability decision also have distinct semantics. With $g = \text{logit}(\rho)$ and $q_{\max} = \max_i q_i$, null wins strictly when

$$1 - \rho > \rho q_{\max} \iff g < -\log q_{\max}. \quad (24)$$

Under uniform competition, null wins when $g < \log K$, so the joint decision depends on both coverage and candidate count.

B.5 Symmetry, Cache Identity, and Numerical Evaluation

With tied maxima, the set of maximizing semantic candidates is invariant. An implementation returning the first maximizing index need not select the same tied member after permutation; identity-based tie-breaking resolves that case. The proofs assume exact arithmetic: floating-point reductions and batching changes can perturb nearly tied scores. Dropout is disabled during inference. A permutation moves complete text, masks, and output metadata; it does not rename labels within the text or rewrite the question. Cache identity requires unchanged encoder and projection weights, tokenizer, templates, length policy, and numeric configuration. Repeated options are gathered repeatedly, preserving multiplicity. Different layers may use different K/V matrices because their outputs still depend only on the same typed option. Invalidating memories after parameter updates is necessary; old training activations are not reusable under the cache-equivalence proposition.

B.6 Detailed Computational Cost

For each uncached layer, the leading multiply–accumulate count is

$$2Kd^2 + 2d^2 + 2Kd + 3df.$$

These terms respectively represent option key/value projections, query and output projections, attention and aggregation, and SwiGLU. Prepared memories remove $2Kd^2$ from the per-request count; the final readout additionally costs Kd . Normalization, softmax, and memory gathering contribute linear work. No K^2 interaction tensor is constructed. Each atomic entry stores $d + 2Ld$ scalar values, with keys evaluated in FP32. Separate assembled-set entries may add storage beyond this atomic-bank count.

B.7 Encoding and Numerical Configuration

The state template is `Decision type: {t}` followed by `Question: {question}` and `State:` with the state on the next line. Each independent option uses `Option for {t}: {option}`. Inputs do not use a chat template. Right padding and the last nonpadding hidden state implement pooling. Length limits are explicit configuration values: the supplied collator can truncate, while full-text evaluations must check lengths before calling it.

The inspected Jeva-4B checkpoint has one matrix block, eight heads of width 96, decision width 768, and SwiGLU width 3072. Its two 2560-to-768 projections and decision core contain 13,370,117 parameters together. Neither initial-state nor final-state normalization is enabled; the FFN uses RMSNorm with $\epsilon = 10^{-6}$. Decision dropout is zero. LoRA uses rank 64, scaling parameter 128, and training dropout 0.05, targeting the configured attention, gated linear-attention, and feed-forward maps. Inference loads the backbone, trained adapter, and custom head together.

Query/key normalization, compatibility scores, candidate softmax, and coverage are computed in FP32, alongside a BF16 backbone. Temperatures are learned through softplus parameterization and initialized to $\tau_{\text{set}} = 0.15$ and $\tau_{\text{dec}} = \tau_{\text{cov}} = 0.10$. The mixing ratio is initialized to $\alpha/\tau_{\text{set}} = 0.60/\sqrt{L}$.

B.8 Type-Specific Losses and Optimization

For Score, let $\bar{y}_i = y_i / \max(\sum_j y_j, 10^{-8})$ be the renormalized candidate target, Q_j, Y_j the cumulative masses sorted by numeric value, and

$R = \max(v_{\text{max}} - v_{\text{min}}, 10^{-8})$. The auxiliary components are

$$\left(\frac{\sum_i (q_i - \bar{y}_i) v_i}{R} \right)^2, \quad \frac{1}{K-1} \sum_{j=1}^{K-1} (Q_j - Y_j)^2.$$

They apply to valid numeric rubrics with $K \geq 2$. An all-null target gives $\bar{y} = 0$; enabled auxiliary terms are still evaluated in the current implementation. For Noul, with $p = q_{i+}$ and $y = \bar{y}_{i+}$, the components are $(p - y)^2$ and, for hard labels, $[m - (2y - 1) \log(p/(1-p))]_+$. The latter clamps p to $[10^{-6}, 1 - 10^{-6}]$ in FP32 only for computing the margin; it does not clamp the reported probability. Soft labels omit the margin.

The trainer clips the accumulated global gradient norm before synchronized optimizer updates, with configured threshold 1.0 in the supplied recipes, and rejects nonfinite losses or gradient norms. Non-reentrant gradient checkpointing can reduce activation memory during LoRA training. In fully frozen-encoder mode the backbone instead stays in evaluation mode and runs without gradients; projections and the decision core remain trainable.

C Detailed Experimental Protocols

C.1 Training curriculum

Jeva follows a staged curriculum with a unified CHOICE, NOUL, and SCORE interface. Training begins with basic typed decisions, intent recognition, and action selection ($\sim 80k$ examples), then expands to preference comparison, semantic judgments, and general reasoning ($\sim 500k$). A counterfactual and verification stage combines $\sim 250k$ additional examples with earlier-data replay, followed by evidence-grounded and open-set decisions ($\sim 60k$). Reading and logical reasoning ($\sim 20k$) precede a balanced mixture of difficult examples and replay ($\sim 60k$). Within this mixture, $\sim 15k$ examples blend original targets with teacher probabilities at a 7:3 ratio, accepting teacher guidance only when its top prediction matches the original target under two independently shuffled candidate orders; 30k replay examples equally mix original targets and preceding-model probabilities. Targeted capability refinement ($\sim 10k$) follows, before training extends to long texts and large candidate pools ($\sim 80k$). A feedback-driven capability-retention phase ($\sim 2k$) precedes and can recur during the final scoring-focused stage with mixed-task replay ($\sim 70k$). Supervision combines hard labels,

soft probabilities, explicit null outcomes, and numerical rubric targets. Reported counts are per-stage training-set sizes and include replay where applicable.

C.2 General evaluation and typed targets

The general suite contains Typed Decisions, Banking77 and MASSIVE at candidate counts $K \in \{8, 16\}$, and three Mind2Web $K = 16$ slices: cross-task, cross-website, and cross-domain. Typed Decisions includes 600 Choice, 600 Noul, and 800 Score examples. Banking77 and MASSIVE include both answer-present and answer-withheld requests. Mind2Web evaluates selection from a supplied candidate pool.

General accuracy uses first-argmax correctness, including null for open-set requests, and is reported for fully supported slices. Unsupported requests are documented separately in the coverage results. Noul Brier measures squared error against the soft positive-candidate target. Score MAE and RMSE compare numerical expectations in the rubric’s original units. Figure 6(a) displays Noul Brier against Score RMSE, showing Jeva’s joint probability and numerical prediction fidelity within the same typed interface.

C.3 Null operating points and candidate permutations

The null comparison uses 3,080 matched Banking77 $K = 16$ requests: 2,492 answer-present and 588 answer-absent cases. Jeva uses its coverage head; the other interfaces use auxiliary coverage Noul outputs or an explicit None candidate.

Candidate-order tests use 600 Choice examples, their original order, its reversal, and up to three seeded random permutations. Outputs are aligned by candidate identity before comparison. Flip rate counts an example whenever any tested order changes its decision; all-order accuracy requires correctness under every tested order. Mean TV is total-variation distance between aligned distributions. Figure 6(c) connects each model’s original and all-order accuracy.

C.4 Large-set support and bounded option encoding

ToolRet contains 500 requests across six distinct-query cohorts, $K \in \{32, 128, 256, 512, 1024, 2048\}$, with 83–84 queries per cohort. Supplied positives are retained;

389 requests are multi-positive. Additional candidates include weak or unjudged negatives, and accuracy accepts a top-scoring target candidate.

Jeva completes all 500 requests and obtains 78.40% aggregate accuracy, including 86.90% at $K = 32$ and 69.88% at $K = 2048$. JevK5 covers the complete sweep through recursive selection and obtains 64.00% aggregate accuracy. Jeva directly scores the full candidate set in a single decision-head pass. The support map in Figure 8(c) reports every measured model–slice combination. Both evaluated InternDecision variants support 65 of 84 requests at $K = 32$ (77.38%); their remaining slices exceed the registered interface limits. The plotted support label 77 rounds this percentage. Accuracy is plotted only for full-support slices.

Request latency is measured on one NVIDIA H200 with device synchronization, excluding model loading. Each model uses its respective runtime; within-model batching and cache comparisons fix the checkpoint and query stream.

C.5 Cache conditions, timing, and numerical agreement

The cache ablation distinguishes an exact-set cache from a hybrid cache that also reuses individual candidate representations. Each episode runs under Off, Set, and Hybrid conditions, with cyclic mode ordering and cache reset between conditions. The completed experiment uses chunk-32/16,384-token batching and contains 171 episodes and 2,188 measured request identities. It includes different states sharing an ordered candidate bank and novel candidate sets with 0%, 50%, or 100% requested option reuse. Novel-set priming uses a different question and candidate set, ensuring zero exact-set hits for the measured request. Priming is timed separately and excluded from reported request latencies. Figure 9(a) shows paired per-request speedup distributions for 46 novel-set requests at each reuse level.

C.6 Input Templates and Typed Examples

Encoder inputs. Jeva separately formats the request and each candidate. The request template $T_x(t, \iota, x)$ is:

```
Decision type: {decision_type}
Question: {question}
State:
{state}
```

The candidate template $T_o(t, o_i)$ is:

```
Option for {decision_type}: {option}
```

Here, `decision_type` is choice, noul, or score. Each formatted text is independently processed by the shared encoder. Candidate inputs contain neither the current request nor other candidates, and the template adds no candidate-position identifier. Consequently, changing the question, state, or candidate order does not change an otherwise identical candidate input.

Choice. Consider the state *I was charged twice.*, the question *Which team should handle this?*, and candidates *Billing*, *Technical support*, and *Delivery*. The request input is:

```
Decision type: choice
Question: Which team should handle this?
State:
I was charged twice.
```

The three candidate inputs are shown below; each line is encoded separately:

```
Option for choice: Billing
Option for choice: Technical support
Option for choice: Delivery
```

A closed-set request selects from q . With `allow_null=true`, selection instead uses $(p_{\perp}, p_1, p_2, p_3)$, where $p_{\perp} = 1 - \rho$ and $p_i = \rho q_i$. Jeva obtains null probability from its coverage gate; no additional None candidate is encoded.

Noul. For the same state, let the question be *Was the customer charged twice?* and set $t = \text{noul}$. The default Boolean candidates are:

```
Option for noul: false
Option for noul: true
```

The interface uses zero-based `positive_index=1`, so the returned Noul value is q_2 in the paper’s one-based notation. It represents the probability of the affirmative candidate, not the probability that the candidate set lacks an answer. The question is included in the request input, while the Boolean candidate representations remain reusable across questions of the same type.

Score. For the question *How severe is the problem?*, set $t = \text{score}$ and provide the rubric:

```
Option for score: Low
Option for score: Medium
Option for score: High
```

With numerical values $(v_1, v_2, v_3) = (0, 1, 2)$, the output is the expectation $\sum_i q_i v_i = q_2 + 2q_3$. The values specify the output scale; they are not appended as an additional encoder field. Likewise, `positive_index` and `allow_null` control output interpretation without entering the text templates. Each typed question produces its own request representation and candidate distribution.

C.7 Evaluation Adapters and Cache Examples

Auxiliary coverage queries. For open-set evaluation, Jev, InternDecision, KEV, Re-Flex, Laya, and Valen receive an auxiliary Noul question alongside candidate selection. It uses the same state and the following instruction template:

```
{question}

Determine whether at least one of the
following candidates correctly applies
to the state:

- {option_1}
- {option_2}
...
- {option_K}
```

The affirmative and negative criteria are:

```
true: At least one supplied candidate
      applies.

false: None of the supplied candidates
      applies.
```

SemiF receives the same coverage instruction as a separate two-option categorical question through its native scoring interface. These adapters introduce no new trained prediction head.

Let $\hat{c}$ be the resulting affirmative probability and $\hat{q}_i$ the conditional candidate probability. The evaluated open-set distribution is

$$\hat{p}_{\perp} = 1 - \hat{c}, \quad \hat{p}_i = \hat{c} \hat{q}_i.$$

Thus candidate selection and answer availability contribute separately to the final prediction.

Explicit null candidates. JevK5 and CLM instead receive an additional candidate appended after the original options. JevK5 uses the literal text None; CLM uses:

```
None of the above. None of the listed
options applies to this request.
```

The appended candidate’s probability is interpreted as null probability. Original candidate texts and their relative order are preserved. For evaluation, all open-set distributions are arranged as $(\hat{p}_{\perp}, \hat{p}_1, \dots, \hat{p}_K)$ and scored by first-argmax, including null. This classification rule is separate from the threshold sweep used to measure null-detection AUROC.

Reuse across new states and sets. Consider two Choice requests using the following candidate identities:

$$\mathcal{O}^{(1)} = (A, B, C),$$

$$\mathcal{O}^{(2)} = (C, A, D),$$

where A , B , C , and D denote *Billing*, *Technical support*, *Delivery*, and *Account security*. These letters identify candidates in the example and are not inserted into their encoder inputs. The first state may report a duplicate charge, while the second reports an undelivered package.

Assume an initially empty cache, fixed model parameters, and sufficient capacity to retain the entries. The first request prepares memory for A , B , and C . The second request misses the exact-set cache because its ordered candidate tuple is new. Hybrid caching nevertheless retrieves the semantic vectors and layer-specific key/value projections of C and A , computes only the missing entry for D , and assembles memory in the new order. Exact-set caching alone recomputes the candidate memory on this miss.

A later request with an already cached candidate tuple can reuse its assembled memory even when its state or question changes. In every case, Jeva still encodes the current request, refines its state, and computes new output probabilities. In document retrieval, different queries reuse the same long passages; in tool selection, requests draw from a shared registry of detailed tool descriptions; in assessment, different responses are evaluated against the same rubric. Caching option-side computation avoids repeatedly encoding this persistent content while preserving request-specific decisions.

Cache identity. Within a fixed model instance, individual-option keys use the formatted token sequences, and exact-set keys use their ordered tuples. Decision type is therefore part of the candidate’s cache identity through T_o . Changes to model weights or encoder settings require cache invalidation. Proposition 3 establishes exact-arithmetic equivalence when cached entries equal their recomputed counterparts.