Sabaki: An Organizational Mixture-of-Experts for Typed Decisions on CPU

Fox3000foxy
Independent Researcher
fox3000foxy@users.noreply.github.com

***Abstract*—Large language models are routinely invoked for tasks that are, at their core, classification: route this ticket, score this lead, flag this message. Sabaki replaces the single-model call with an *organizational* mixture-of-experts: a keyword- and language-aware router dispatches each input to a small, independently trained and versioned classifier plugin (nano TF-IDF, < 1 ms; mini MiniLM ONNX, $\sim$20 ms on CPU; or a cross-encoder fallback), which returns a typed answer—choice, score, or yes/no—with a calibrated confidence. Specialists train in minutes on thousands of labeled rows under a strict held-out protocol; failures are published alongside wins. On the public typed-decisions split, Sabaki scores 0.61 (nano) and 0.69 (mini) against 0.75–0.80 for models needing a GPU or an API, locally, with no API key: a few points for millisecond CPU latency instead of API round-trips (competitor latencies were not measured; treat the gap as architectural, not metered). Routing adds nothing measurable over the fallback alone (0.845 vs 0.865 end-to-end) while the flat nano ties everything at 0.89: the value proposition is modularity, auditability, and the cheapest competent model answering first. We report seven days of development across twelve repositories, including negative results (int8 cross-encoders, distillation, Platt scaling) and the bugs that shaped the protocol.**

***Index Terms*—mixture of experts, text classification, CPU inference, model routing, multilingual NLP, prompt injection detection**

I. Introduction

A growing share of language-model API calls are not open-ended generation but *decisions*: which bucket, what severity, how urgent, yes or no. Paying autoregressive prices—latency, dollars, data egress—for a label is wasteful, and a giant model invoked for a label is unauditable: the same weights answer everything, so nothing can be versioned, rolled back, or evaluated per skill.

Sabaki takes the opposite route. It is a TypeScript router plus a registry of *specialist plugins*. Each plugin owns one domain (e-mail triage, phishing, jailbreak, game navigation), declares the questions it answers in a manifest, and ships its weights bundled (TF-IDF JSON for nano, ONNX for mini). A router scores every plugin on the input text and delegates to the winner; when no specialist clears its threshold, a cross-encoder generalist answers as fallback. The whole system serves a Jev-compatible HTTP endpoint (`POST /v1/systemone`) and a batch CLI (`bunkitsu resolve`) that resolves whole CSV files row by row.

Three design choices define the project. **First**, experts are *organizational*, not neural: independent repositories, versions, and releases, so a phishing fix ships without touching e-mail triage. **Second**, evaluation is held-out by construction: eval rows are generated or split *before* training and never enter the mix, and every reported number in this paper comes from those splits. **Third**, negative results are published: failed distillations, a Platt scaling that hurt, and two data-loader bugs are documented in §XI because they changed the protocol.

The system went from an empty organization to a served API, two model families, eight specialist plugins (six workflow plus two security), a 16-language track, and a batch CLI in seven days (Oct. 13–19, 2026); §V reconstructs the timeline from commit histories.

Our contributions are: (i) an organizational MoE design where routing, versioning, and serving are decoupled per skill; (ii) a held-out-first training protocol with hash-verified leakage audits; (iii) a 16-language extension with a strict non-Latin fallback perimeter, honestly evaluated; (iv) a batch resolution tool that makes every decision traceable; and (v) a full negative-results record, including two data-loader bugs that silently zeroed a 60-class task.

II. Architecture

A. Router and plugin contract

The router (`Sabaki` class in the `sabaki` core package) scores each registered plugin on the raw input state. The base score is lexical: hits$/\sqrt{|\text{keywords}|}$ with loose inclusion in both directions, gated by a per-manifest threshold (default 0.15). A plugin manifest declares `name`, `domain`, `version`, `questions`, `primary`, `keywords`, an optional `threshold`, and, since v1.9.0, optional `languages` (ISO codes). When several plugins share a domain, the language filter disambiguates: a Latin-script phishing text routes to the Latin mini, a Chinese one to the multilingual backend. Inputs too short for reliable detection skip the filter and compete normally. The same `route()` call powers interactive serving, batch resolution, and the live demos.

A plugin is any object with {`manifest, classify`}; async `classifyAsync` variants are awaited by `decideAsync`. The contract fits in one screen, deliberately: a new specialist is a new repository, not a core change.

Routing details matter. Scores are hits$/\sqrt{|\text{keywords}|}$ with loose substring matching in both directions, so "verify" catches "verification" without stemming. A manifest threshold (default 0.15, tunable per plugin) gates delegation; below it the input is `unknown` and the fallback answers. Two incidents shaped these defaults: a `threshold: 0` copied from the

generalist turned every plugin into a catch-all (first registered won everything), and a 49-keyword manifest diluted single-hit scores under the default, fixed by a per-manifest 0.1. Language filtering composes with scoring rather than replacing it: declared languages only *exclude*, they never elect, so a benign German weather query still falls through to the fallback instead of being forced into phishing.

B. Three execution tiers

Nano bundles TF-IDF + logistic regression weights (a few hundred kilobytes) and runs with zero dependencies in under a millisecond, in pure TypeScript with exact scikit-learn parity (sublinear TF, smoothed IDF, L2 norm). *Mini* loads an ONNX MiniLM encoder plus a linear head through `onnxruntime-node` (∼87 MB fp32, ∼22 MB int8) in about 20 ms per request on a desktop CPU. The published int8 minis are dynamically quantized; cross-encoders are kept in fp32 because int8 cost them 0.13 accuracy with no speed gain (§XI).

The nano tier deserves its precision note: the TypeScript implementation reproduces scikit-learn exactly (sublinear TF, smoothed IDF, L2 norm, $[a-z]3$, unigrams+bigrams), verified prediction-for-prediction against Python on the 20-question suite, so a weights JSON trained in Python decides identically in the browser. The WordPiece runtime likewise matches Hugging Face tokenization id for id, including the Unigram Viterbi path added for the multilingual base (SentencePiece scores, byte-fallback to unk). Parity is not a claim but a test: `fun_test.py` and the ONNX parity bench run it on every export.

The *generalist* tier is a MiniLM cross-encoder (question [SEP] option [SEP] state, one forward per option, argmax over softmax). The served v1.0 mini (22 M parameters, from scratch) answers any runtime-defined question; the multilingual v8 (paraphrase-multilingual-MiniLM-L12-v2, 117 M) serves as the non-Latin fallback, dispatched per language. A lazily-loaded `modelDir` plus `SABAKI_*_DIR` overrides make every weight swappable without code changes.

C. Calibration, abstention, batch

Mini logits pass through per-question temperature scaling fit on held-out data. A `confidence` in $[0, 1]$ accompanies every decision, and callers can set `abstainAt` to escalate below a threshold instead of acting. `bunkitsu resolve` applies the full pipeline to CSV files (`state,question,type,options` → `decision,confidence,probs,domain,model`), giving every row a traceable model identity—the unit of evaluation in §VI is always the held-out row.

III. Training and Evaluation Protocol

Every specialist follows the same pipeline: *harvest* (public datasets plus natively generated rows), *validate* (schema, label-in-options, exact-duplicate removal, now label-aware after a dedup bug kept the first of two conflicting labels), *mix* (shuffled, seed-fixed, eval rows excluded by construction),

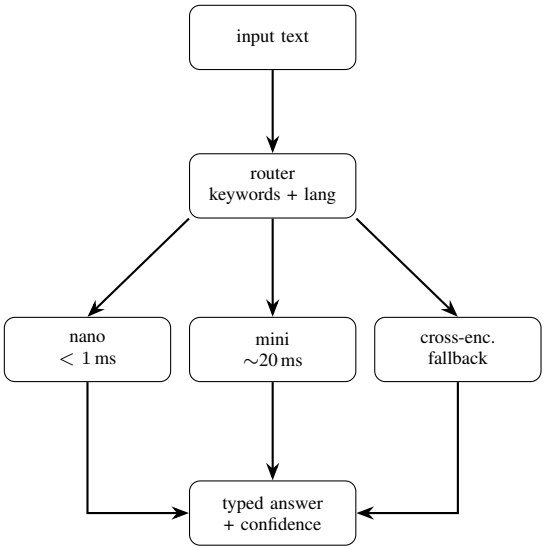


Fig. 1. One input, one winner: routing, three tiers, typed answer.

train (cross-entropy over options with negative sampling for wide option sets), and *eval* on the held-out splits.

Two protocol rules proved load-bearing. First, eval rows are produced *before* or *independently of* training rows (separate generation batches, upstream test splits) and a hash-overlap check (0/1600 on the phishing track) guards every release. Second, sampling is explicit: rows with more than eight options train on gold plus seven random distractors, and the train-time evaluator indexes the *sampled* options—both fixes for bugs that silently zeroed 60-class intent accuracy (§XI).

Training uses AdamW (lr 2×10^{-5}), mixed precision on T4s, and data-parallel pairs; a 22 M binary mini converges in minutes on ∼10k rows while a 117 M 3-epoch multilingual run takes about six hours. Checkpoints upload per epoch to a private hub repository, so killed runs resume instead of restarting. Calibration uses temperature scaling only. A Platt (vector-scaling) fit improved training ECE but degraded held-out ECE almost everywhere (e.g. customer-service `action` 0.155→0.371) and was never shipped; served minis carry temperatures, not Platt parameters.

A. Routing ablations

Threshold and keywords interact plainly. With 43 keywords a single hit scores $1/\sqrt{43} \approx 0.15$, exactly the default gate: the French phishing SMS routes, a weather question does not. Growing the keyword list without also tuning the threshold dilutes every hit, a mistake caught live when German phishing fell through to the fallback and fixed by per-manifest thresholds plus language priors. Short subjects (under four letters after normalization) skip language filtering entirely rather than route on noise.

B. Nano versus mini

The two tiers differ in kind, not just size. Nano is synchronous pure TypeScript ($<$ 1 ms, hundreds of kilobytes, zero dependencies): TF-IDF vectorize, dot product, softmax.

Mini is asynchronous ONNX (∼20 ms, 22–90 MB): a MiniLM encoder plus a linear head per question, with temperature calibration. `autoClassifier` bridges them: sync calls always use nano, async calls lazily upgrade to mini on first use and fall back to nano offline, so routing never changes—only inference quality does. Table VIII shows the gap is task-dependent (agent-trace 0.59 → 0.82, invoice nearly tied), which is why both ship in every plugin.

C. Inference optimization

Three changes, all measured, none touching weights. Batching options into one ONNX forward (padded to max length, masked) replaced one `session.run` per option: 2 options decide in 6.2 ms (was ∼20 ms published) and 8 in 16 ms on desktop CPU, with identical decisions. Language detection runs only when a registered plugin declares `languages`, so keyword-only setups pay nothing. TF-IDF featurization touches only present terms instead of sweeping 3 000 dense dims. Batch resolution parallelizes rows across a bounded worker pool (default 4, order preserved) since rows are independent given shared cached sessions.

D. Standalone routing accuracy

Measured per label (60 attack rows should route to the specialist, 60 benign rows should not), the router is precise but quiet: phishing recall 0.17 at FPR 0.03, jailbreak recall 0.12 at FPR 0.02, against customer-service at 1.00 on jargon-heavy states. Most attacks therefore reach the generalist fallback, which is why end-to-end answers stay correct while specialists sit idle. Routing, not inference, is the system's weakest link, and the semantic-centroid attempt (§XI) did not fix it. This subsection exists because the component that names the architecture deserves its own evaluation.

E. Worked example

Consider `''SMS: régularisez 96 EUR via ce lien''` with `suspicious?`. Token overlap gives the phishing plugin two hits over 43 keywords (0.30), above its 0.1 threshold; the French `langid` head agrees (Latin, fr, 0.94). The cross-encoder scores `yes` at 0.91 and the row resolves with full provenance. A weather question on the same text scores zero everywhere and falls through to the mini generalist. Nothing in this path needs a GPU, a download, or a key: the weights rode along with the plugin.

IV. The Jev Contract

Every decision, interactive or batched, speaks one protocol: a state, a typed question, and options or criteria. `choice` returns the winning option with full probabilities; `noul` (none-of-the-above) abstains by construction; `score` maps onto levels. Names are mapped, never guessed: a manifest declares `aliases.questions` (paraphrases of its own question) and `aliases.options` (synonyms of its native labels). The router matches the requested question onto each candidate's native questions by exact id, declared paraphrase, else primary, so models never see unknown heads, and projects the native decision back onto the caller's option names. When the two vocabularies are disjoint—a caller asking {`clean, danger`} of a {`legit, phishing`} specialist—the specialist is skipped rather than allowed to return a label the caller never offered, and the fallback, which scores each option as text, answers instead, with probabilities renormalized over the caller's options. The answer carries the responder's model id for traceability. Bunkitsu is this contract applied to rows: same routing, same calibration, plus `domain` and `model` columns so no decision is anonymous.

V. Timeline: Seven Days, Twelve Repositories

Table I reconstructs the build from commit histories of the `sabaki-js` organization (plus one personal repository). The pattern is consistent: skeleton day one, real classifier the same day, stable release within 48 h, bundled offline weights at the end.

TABLE I
Build timeline (Oct. 13–19, 2026). Tags are GitHub releases.

Date	Milestone
Oct 13	Core router + 4 workflow skeletons; nano TF-IDF; mini ONNX runtime (17 ms); v1.0.0; trusted publishing
Oct 13	5 specialist plugins + mail + doom skeletons
Oct 14	Doom nano + video demo (MAP01, 36 s); ONNX serve
Oct 15	Jev answers; generalist in core; site v2/v3; mail mini
Oct 16	Generalist v0.8; versions.json pointers; `Sabaki.serve()`
Oct 17	Generalist-mini **v1.0.0 served** (A0, R=0.616 phishing)
Oct 18	Bundled offline weights (all plugins); multilingual corpus (193k rows, 16 langs); jailbreak specialist (0.940)
Oct 19	Language router + nano langid; phishing-latin (0.94); bunkitsu batch CLI + site page; core v1.10.x

VI. Results

All numbers below are held-out: the typed-decisions public test split, or eval files excluded from training before it started.

TABLE II
Typed-decisions (public test split) and latency.

System	Accuracy	Latency
OpenDecider-nano	0.796	GPU/API
Laya typed-decisions	0.766	GPU/API
TypeSafe Jev	0.754	API
sabaki mini	**0.69**	**∼20 ms CPU**
sabaki nano	**0.61**	**<1 ms CPU**
Laya base	0.362	GPU

Paired nano vs forced mini: McNemar $\chi^2 = 1.93$ (26 vs 16 discordant), not significant—the whole table is one statistical tie, and the cheapest model is the rational default.

Table II sets the trade: a few accuracy points for millisecond CPU latency instead of API round-trips (competitor latencies not metered—see Appendix A). The specialists add the long tail the benchmark misses. Paired on the same 2 000 workflow-question items, mini beats nano significantly

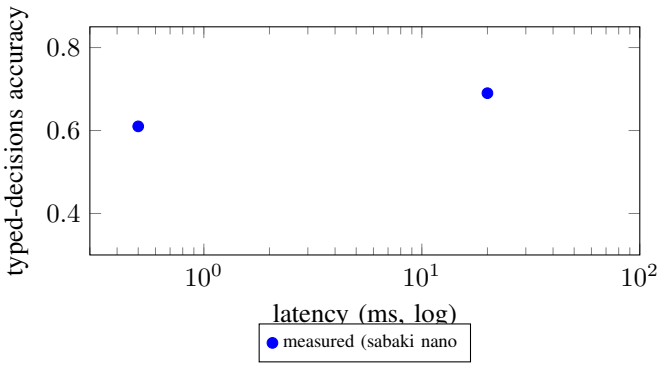


Fig. 2. Our measured points only. Competitor accuracies are in Table II; their latencies were never metered and are not plotted.

TABLE III
END-TO-END ON FR+EN PHISHING ($n = 200$, SAME ROWS). THE FLAT NANO IS THE PHISHING-ONLY SPECIALIST WITHOUT ROUTING: IT TESTS ROUTING AGAINST SPECIALIST-ALONE, NOT AGAINST A FLAT-EVERYTHING MODEL.

Configuration	Accuracy
Flat nano (phishing-only, no routing)	0.890
Fallback alone (generalist-mini)	0.865
Full pipeline (route + specialist/fallback)	0.845
Forced specialist (mini, no routing)	0.840

TABLE IV
RISK-COVERAGE ON FR+EN PHISHING ($n = 200$, FORCED MINI SPECIALIST): ACCURACY OF THE TOP-CONFIDENCE FRACTION (ABSTAIN THE REST).

Coverage	Accuracy	Threshold
1.00	0.840	—
0.90	0.883	0.92
0.80	0.919	0.99
0.70	0.943	1.00
0.50	0.990	1.00

TABLE V
PHISHING-LATIN MINI (22 M, 7 LANGUAGES, 100 HELD-OUT EACH).

Lang	Acc	P	R	F1
de	0.97	0.96	0.98	0.97
es	0.99	1.00	0.98	0.99
id	1.00	1.00	1.00	1.00
tr	1.00	1.00	1.00	1.00
vi	0.93	0.88	1.00	0.94
en	0.85	0.97	0.72	0.83
fr	0.83	0.79	0.90	0.86

(McNemar $\chi^2 = 30.35$, 147 nano-only vs 259 mini-only wins), so the tier gap is real even where Wilson intervals touch. Against TypeSafe Jev (0.754, sample size unpublished), our 0.69 [0.62–0.75] may sit in noise—stated, not hidden. Phishing-latin (22 M mini, 7 languages, 100 held-out cases each): de 0.97, es 0.99, id/tr 1.00, vi 0.93, en 0.85, fr 0.83. Its nano sibling (TF-IDF, trained locally in minutes on the same mix) reaches 0.884 overall on the same splits—French even above the mini (0.89 vs 0.83)—so the released plugin ships both: sync nano under a millisecond by default, async mini on demand. Anti-jailbreak (EN+FR, 200+200 held-out): 0.940 overall (EN 0.925, FR 0.955), including natively generated French attacks after two generation agents refused and the gap was filled by translating public attack corpora. Multilingual v8 (117 M, non-Latin fallback): 60-class intents 0.90 (FR) / 1.00 (EN) on held-out MASSIVE eval, non-Latin phishing at 1.0, but Latin phishing recall 0.736 and typed-decisions 0.31—hence its strict fallback perimeter. Language ID itself (scripts exact, 9 Latin languages TF-IDF, 316 KB) scores 16/16 on natural sentences. Two gaps are within noise and one is not: nano-vs-mini French (0.89 vs 0.83, n=100 each) overlaps under Wilson intervals, so no claim follows; the v8 phishing drop (macro-R $0.861 \rightarrow 0.736$, n=1600) is significant. Our hypothesis—stated as such, the ablation is not run—is capacity absorption: precision stays 1.0 while recall falls, the signature of added hard benigns plus a 60-class head competing for one scorer.

VII. TWO SPECIALISTS UP CLOSE

Mail began as 162 hand-labeled messages (bucket accuracy 0.848) and grew into a 4 126-message Takeout corpus with anonymized PII, eight buckets, and a 60-keyword router; nano reaches 0.865, mini 0.889. The suspicious question stayed experimental (three examples, recall 0/3 held out) until the phishing track gave it real data—an honest example of a documented weak head later reinforced by a sibling project.

Doom is the outlier that validates the contract (see the video demo): text states from the engine, no vision, a Python driver executing over HTTP, three questions (`command/target/weapon`), a 15-tick latch on `open_door`, and a full MAP01 exit in 36.4 s versus a 61 s reference. Nothing in the router knows about games: one domain, three questions, thresholds—the same endpoint sorts mail and fights demons. Throughput benches (nano vs mini, local vs Kaggle) showed the mini at ∼20 ms/state, fast enough to close the loop in real time.

VIII. CORPUS CONSTRUCTION

The multilingual track harvests three public sources per locale (MASSIVE train, sentiment tweets train, native generated phishing) plus English legacy mixes and French Allocine reviews, for 193 k training rows (EN 40.9%, FR 25.9%, other 35.2%) and 18.5 k held-out rows (upstream test splits or separate generations, never mixed into training). Validation enforces schema, label-in-options, and exact-duplicate removal; an overlap audit (0/1600 eval states found in training)

guards every release. The jailbreak track (10 k rows EN+FR) merges public attack corpora with natively generated French personas after two generation agents refused; translation of public attacks filled the gap. Language ID trains on 1 500 rows per Latin language with a 9-way TF-IDF head (3 000 features, 316 KB) and reaches 0.927 held-out, 16/16 on probe sentences with script shortcuts.

IX. Multilingual Corpus Budgets

Scale is deliberately uneven (details and per-locale volumes in §VIII, Table VI): the imbalance is a feature under test, not an accident. Latin scripts share the mini, others fall back to the 117 M cross-encoder, and per-language reporting keeps the trade visible instead of averaged away.

TABLE VI
MULTILINGUAL CORPUS VOLUMES (ROWS; EVAL HELD OUT SEPARATELY).

Source	Train	Eval
MASSIVE ×14 locales	3 500 each	500 each
Tweets ×6 langs	1 800 each	full test split
Generated phishing ×14	300 each	100 each
Allocine FR (reviews)	40 000	2 000
Legacy EN mix (v4)	75 000	—

X. Batch Resolution with Bunkitsu

`bunkitsu resolve` maps the interactive pipeline onto files: each CSV row is routed and decided exactly as the API would, and the output appends `decision, confidence, probs, domain, model`. The `model` column names the deciding backend (down to the plugin version), making every row auditable. Rainbow-colored, comma-aligned examples on the showcase page double as the format specification.

TABLE VII
ANTI-JAILBREAK MINI (EN+FR, 200 HELD-OUT EACH).

Set	Acc	P	R	F1
EN (n=200)	0.925	0.921	0.930	0.925
FR (n=200)	0.955	0.925	0.990	0.957
All (n=400)	0.940	0.923	0.960	0.941

XI. Negative Results

int8 cross-encoders: −0.13 accuracy, zero speedup—rejected; heads quantize, cross-encoders stay fp32. *Distillation v1* (0.505 student vs 0.69 specialist): not pursued. *Platt scaling*: see §III; temperature only. *Option cap*: truncating 60-option rows to the first eight forced the label to index 0 and taught the model to always answer the first option—fixed by gold-plus-seven negative sampling. *Eval indexing*:

TABLE VIII
SPECIALIST PLUGINS (HELD-OUT; NANO / MINI).

Plugin	Nano	Mini
customer-service	0.900	0.91
invoice-processing	0.890	0.89
security-incidents	0.730	0.76
agent-trace	0.590	0.82
mail (bucket)	0.865	0.889
doom nav	—	MAP01 exit
phishing-latin	0.88	0.94 avg
jailbreak EN+FR	—	0.94

TABLE IX
MULTILINGUAL V8 VS V7 (HELD-OUT; INTENTS = 60-CLASS PROBE).

Track	v7 (buggy)	v8 (fixed)
Intents FR / EN	0.20 / 0.10	0.90 / 1.00
Phishing macro-R (16 langs)	0.861	0.736
Non-Latin phishing	∼1.0	∼1.0
Typed-decisions	—	0.31

TABLE X
CPU LATENCY PER DECISION (DESKTOP CPU, ONNX).

Tier	p50	Notes
Nano (TF-IDF)	<1 ms	zero deps
Mini int8 (Head)	∼20 ms	22 MB
Cross-encoder fp32	∼30 ms	1 fwd/option
Multilingual 117 M	∼100 ms	fallback only

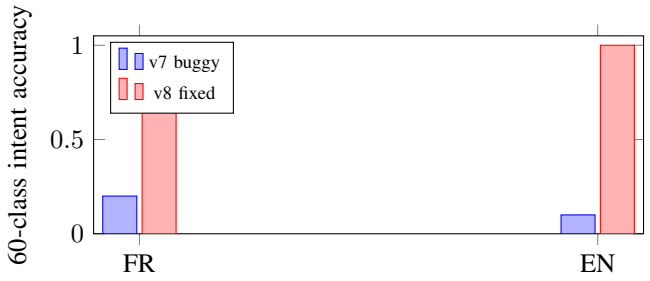


Fig. 3. Negative sampling unblocks 60-class intents (held-out MASSIVE probe).

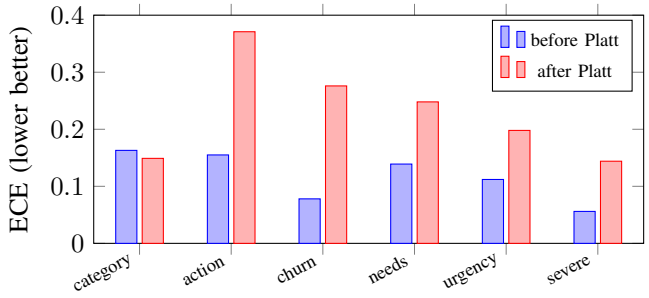


Fig. 4. Platt scaling overfits: held-out ECE degrades almost everywhere.

predictions over shuffled options were mapped back onto fixed order, reporting 0.00 on intents even as weights learned; fixed by threading sampled options through. The honest end-to-end picture on FR+EN phishing ($n = 200$, same rows) is humbling: full pipeline 0.845, fallback alone 0.865, forced specialist 0.840, flat nano 0.89. Routing adds nothing measurable here; the cheapest model wins. The architecture's value on this slice is modularity and auditability, not accuracy—stated plainly because the numbers leave no other reading. *Threshold 0*: a catch-all copied from the generalist made the first plugin win everything; restored to 0.15 defaults with per-manifest tuning.

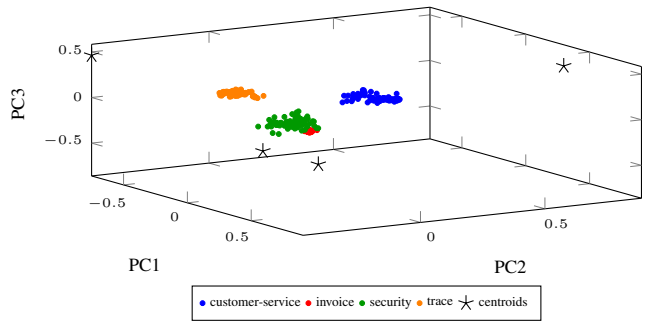


Fig. 5. MiniLM states, PCA to 3D (63.4% variance): domain clouds overlap, centroids (stars) sit close—why cosine routing stalls at 0.289 while keywords discriminate.

Semantic router v2: domain centroids μ_d (mean of L2-normalized MiniLM embeddings of train states) with argmax cosine routing and a tuned threshold scored 0.289 absolute on the hard routing set—far below keywords—and was parked. Embeddings blur exactly where keywords discriminate (short subjects, jargon), so v2 stays a prototype until a better encoder or supervised centroids earn their place. *Semantic question mapping*: declared aliases only cover wordings somebody wrote down. To widen the contract, a multilingual bi-encoder (`paraphrase-multilingual-MiniLM-L12-v2`, 117 M, Apache-2.0, int8) was tried as a second mapping layer: it embeds the caller's wording and each plugin's native question description, and the routed plugin may answer when its own description wins the cross-plugin competition (threshold 0.50; the floor is 0.45, where false positives jump to 17%). Out-of-domain wordings map cleanly, and on 54 hand-written cases (EN/FR/ES/DE/IT/PT/RU/JA/ZH) the layer raised the positive wordings served by a specialist from 11/28 to 27/28 (Table XI). It also handed 3/26 negatives to the wrong specialist (12%): "ce message est-il bien redige ?" scores 0.608 against the phishing description, "quel est le ton de ce message ?" 0.564, above the threshold and above several true positives (0.50–0.67). Subtracting the similarity to generic questions about the same object removes those false positives only by collapsing true positives to a 0.04 margin. The failure is systematic and reproducible, not noisy: a bi-encoder ranks topic, not intent, so any question sharing the entity ("this message") scores high. The layer ships opt-in and disabled, since +57 points of recall against 12% wrong answers fails the rule that abstaining beats answering wrongly. An earlier evaluation of the same layer reported 0/23 false positives; that number was an artifact of a negative set built from out-of-domain questions only, and is superseded here. Variants hand-picked for the probe (96% recall) gave way to variants derived from the manifests, which score 82% at the same 0.52 operating point: the cheaper, less flattering construction was kept. The plumbing (per-question variants, cached embeddings, cross-plugin competition) is kept for a cross-encoder trained on (query, native question) pairs; a per-plugin threshold alone cannot see cross-plugin confusion, accepting "is this a scam?" as a jailbreak question at 0.57. The by-product is positive: unlike the cross-encoders above, bi-encoder int8 quantization is free (471 MB → 119 MB, 11.2 ms → 4.3 ms, identical decisions at every threshold). *Template-grown data*: to fix French hard phishing, 142 combinatorial rows (generic heads × tails) joined 600 agent-written French rows. Probes hit 0.98—but the templates were near-duplicates of training, so held-out French *fell* (0.88 → 0.72) while English recall slid. Synthetic regularity taught patterns, not the boundary; the shipped weights stayed v1 and the lesson joined the protocol: no template-grown rows without a held-out delta to prove them. *versions.json on* `main`: the pointer file was fetched from a branch that never existed (404, silent fallback), so no version update ever took effect until the URL moved to `master`.

TABLE XI
SEMANTIC QUESTION MAPPING, 54 HAND-WRITTEN CASES, BI-ENCODER INT8, THRESHOLD 0.50. A POSITIVE IS A WORDING THE ROUTED SPECIALIST SHOULD SERVE; A NEGATIVE IS A QUESTION ABOUT THE SAME STATE THAT IT SHOULD NOT. THE LAYER IS DISABLED BY DEFAULT.

Mapping	Positives served	Negatives served
Declared aliases only	11/28	0/26
+ bi-encoder layer	27/28	3/26

XII. Error Analysis

Three failure families recur. *Nested negation* ("refund if not delivered unless cancelled but not when the address was wrong") drops confidence to ∼0.68 and still answers instead of abstaining; `abstainAt` exists but is opt-in, so production

TABLE XII
PLATT SCALING ECE BEFORE → AFTER (HELD-OUT; LOWER BETTER). SHIPPED: NEITHER (TEMPERATURE ONLY).

Question	Before	After
category	0.163	0.149
action	0.155	0.371
churn_risk	0.078	0.276
needs_human	0.139	0.248
urgency	0.112	0.198
discrepancy	0.056	0.144

leans on thresholds. Table IV shows the mechanism works when used (0.84 → 0.99 from full to half coverage), which is the evidence for switching the default on. For flag-style decisions this is a genuine deployment risk we state plainly: confidence without abstention by default can automate a wrong answer, and operators should gate on confidence until the hard-data iterations land. *Hard French phishing* ("your account will be blocked") reads as legitimate urgency to the Latin mini (recall 0.90 FR); the generalist fallback catches some of these, which is why routing keeps it in the loop. *Latin-script orphans*: Italian (F1 0.29) and Portuguese (0.60) lack the data volume of their siblings despite sharing the script—a reminder that script is necessary but not sufficient for transfer.

XIII. DEPLOYMENT

One API serves everything (`pm2`, port 8001): specialists preempt by routing, the generalist fallback answers the rest, now dispatched by language (Latin → mini v1.0, else multilingual v8). Weights resolve through `ensure*` helpers (HF release assets, SHA256-verified, 24 h-cached versions pointer; Section XI tells how that pointer once pointed nowhere) or ship bundled in plugin repositories; the 470 MB multilingual weights cannot be bundled (GitHub's 100 MB file limit) and stay download-only. Hardcoded fallbacks cover network failure, and every paper number re-derives from a circuits-CSV run. A batch CLI (`bunkitsu resolve`) and a static showcase site with live API demo complete the loop from training row to served decision. Serving is deliberately boring: one `pm2` process on the API host, four workflow minis preloaded, security cross-encoders lazy on first request (470 MB multilingual included), and a Jev-compatible `/v1/systemone` surface identical from nano to fallback.

XIV. SERVING HONESTIES

Three serving details deserve print because they are usually hidden. First, token accounting is a heuristic: `input_tokens` is `state.length >> 2`, stated as-is in the response, not a tokenizer count. Second, the core ships dependency-free: ONNX runtime is dynamically imported, so nano users never install it and a missing backend throws a one-line install hint instead of a stack trace. Third, multi-question requests name every responder: when two questions hit two backends the model field reads `sabaki-routed(a,b)`, so mixed provenance is visible rather than averaged. Confidence rounds to 10^{-3}; temperatures apply as x/T with optional per-class bias, read straight from the model directory's `model.json`.

XV. RELATED WORK

Mixture-of-experts usually means gated neural blocks inside one model; Sabaki's experts are repositories with releases, closer in spirit to Unix pipelines and model zoos than to Switch Transformers. The Jev protocol (typed questions over HTTP) standardizes what OpenDecider, TypeSafe Jev, and Laya each do ad hoc. Multilingual intent benchmarks (MASSIVE, 60 classes, 16 locales here) and jailbreak benchmarks (JailbreakBench) supply the held-out judges; MLCommons-style safety taxonomies inform the jailbreak label set. On the serving side, Jev-compatible HTTP endpoints (as in TypeSafe Jev) let any client swap Sabaki in without SDK changes; `bunkitsu` extends the same contract to files. Unlike distillation pipelines that compress one giant model, Sabaki never had a giant model: capacity arrives as new repositories. A note on the title's "MoE": it means routing to versioned classifiers, not gated neural blocks; the qualifier "organizational" carries that distinction, but readers expecting Switch-style machinery should recalibrate here. Likewise the honest baseline gap: one fine-tuned classifier over everything, FastText, SetFit, and a small quantized LLM were not run, and would sharpen the trade analysis—listed, not hidden. For calibration, the project follows the temperature-scaling literature and stops there: the vector-scaling experiment in §XVII is, to our knowledge, a rare published Platt failure with per-question ECE tables, kept precisely because silent trimming of negative results is how benchmarks rot. For multilinguality, MASSIVE supplies the intents and OPUS-MT the translation pivots; the contribution is not data but the fallback perimeter—knowing exactly where the small model stops and the big one starts, per language, measured, on the record.

XVI. GENERALIST GENERATIONS

Five generations trace the cross-encoder line: v4 (mix 0.848, phishing R=0.026), v5 (0.838 / 0.066), A (22 M, 80k rows, 0.898 / 0.406), mini v1.0.0 served (0.756 / R=0.616 on real phishing, 0.536 typed-decisions), B (144 M, 0.692 / 0.062, rejected: capacity without data). The 144 M prototype confirmed that parameters do not substitute for supervision—a result that killed the TPU port the same week (2×T4 sufficed; XLA risk unjustified).

XVII. CALIBRATION IN DETAIL

Every mini ships per-question temperatures fit on held-out data (same head, divided logits, ECE 0.06–0.17 across workflows, Table XII). Vector scaling (per-class bias plus temperature) won on the fit split and lost almost everywhere held-out (Fig. 4), a textbook overfit: a one-sixth held-out slice per workflow with 10-bin ECE cannot support $K+1$ parameters per question, so even the failure itself is noisy—and still informative. The shipped rule is therefore minimal:

one temperature per question, frozen at release, re-fit only on new held-out data. The showcase site states ECE below 0.16 for the shipped temperature fits (worst measured head: `action` at 0.151, $n = 100$); Table XII spans 0.06–0.17 because its Before column is raw logits, not the shipped model—same data, different stage, stated once here.

XVIII. Threats to Validity

Held-out evals share generators with training (same agent families, same upstream corpora), so absolute scores are optimistic proxies; only *relative* comparisons (v7 vs v8, mini vs baselines) and the public typed-decisions split are fully external. Confidence is calibrated on near-distribution data and degrades on twisty inputs (§XII). Latency figures are desktop-CPU single-stream. On ethics: the jailbreak track trains a *detector*, not an attacker, on already-public corpora, translated where agents refused to generate; we publish the detector, not new attacks.

XIX. Limitations and Future Work

French hard phishing (recall 0.90→), Italian and Portuguese phishing, and nested-negation logic (confidence drops, no abstention by default) are the known gaps, each with data already harvested for the next iteration. Distilling the 117 M multilingual intents into a 22 M mini, and per-question temperature recalibration, are open, in order: (i) hard French phishing (the account-blocked template that beats recall 0.90), (ii) Italian and Portuguese volume, (iii) nested-negation logic with abstention on by default, (iv) a 22 M distillation of the multilingual intents, (v) and, if question mapping is to leave opt-in, a cross-encoder trained for it rather than a bi-encoder (§XI). A Doom agent already plays MAP01 end-to-end through the same endpoint (text states from the engine, no vision), proving the router handles sequential decisions, not just single shots. The showcase site publishes every number above—including the weak ones—with protocols, so the paper's tables are a subset of a living document, not a frozen claim. The template repository now encodes the bundled-weight pattern so the ninth plugin costs as little as the eighth.

XX. Conclusion and Outlook

Seven days of commits say more than the tables: small local classifiers plus a fallback plus transparency beat leaderboard climbing, and published failures (Platt, int8, silent index bugs, embedding-based question mapping) compound into protocol. The remaining work is data, not architecture (§XIX). The ninth plugin will cost less than the eighth; that curve, more than any score, is the point. In short, Sabaki shows that a useful share of "LLM work" is routing plus tiny classifiers: versioned, evaluated on held-out splits, served on CPU in milliseconds, with failures published next to wins. The organizational MoE scales by adding repositories, not parameters.

References

[1] S. FitzGerald et al., "MASSIVE: A multilingual dataset for intent classification," 2022.
[2] P. Chao et al., "JailbreakBench: An open robustness benchmark for jailbreaking large language models," NeurIPS, 2024.
[3] J. Tiedemann and S. Thottingal, "OPUS-MT: Building open translation services for the world," EAMT, 2020.
[4] W. Fedus et al., "Switch Transformers: Scaling to trillion parameter models," JMLR, 2022.
[5] W. Wang et al., "MiniLM: Deep self-attention distillation," NeurIPS, 2020.
[6] N. Reimers and I. Gurevych, "Sentence-BERT," EMNLP, 2019.

Appendix

TABLE XIII
The twelve repositories (all MIT, all public).

Repository	First commit	Head tag
sabaki (core)	Oct 13	v1.10.2
sabaki-models	Oct 13	versions.json
plugin-generalist	Oct 15	v0.1.1
plugin-template	Oct 13	v1.1.0
plugin-customer-service	Oct 13	v1.0.2
plugin-invoice-processing	Oct 13	v1.0.2
plugin-security-incidents	Oct 13	v1.0.2
plugin-mail	Oct 13	v1.1.2
plugin-agent-trace	Oct 13	v1.0.2
plugin-doom	Oct 13	v1.0.1
plugin-phishing-detection	Oct 19	v1.1.0
jailbreak-detection	Oct 18	v1.1.4

Routing. For plugin p with keywords K_p and input words $W(x)$,

$$\mathrm{hits}(p,x) = |\{k \in K_p : \exists w \in W(x),\, w \subseteq k \vee k \subseteq w\}| \quad (1)$$

$$s(p,x) = \frac{\mathrm{hits}(p,x)}{\sqrt{|K_p|}}, \quad \mathrm{eligible}(p,x) = \big(L_p = \emptyset \vee \mathrm{lang}(x) \in L_p\big). \quad (2)$$

Delegate to $\arg\max_p s(p,x)$ iff $s \geq \tau_p$ ($\tau = 0.15$ default); else `unknown` goes to the fallback.

TF-IDF (nano). With $c(t,d)$ the count of term t in d, document frequency $\mathrm{df}(t)$ over N rows:

$$\mathrm{tf}(t,d) = 1 + \ln c(t,d), \quad \mathrm{idf}(t) = \ln \frac{1+N}{1+\mathrm{df}(t)} + 1, \quad \mathbf{x}_d = \frac{\mathbf{v}_d}{\|\mathbf{v}_d\|_2}. \quad (3)$$

Terms are $[a-z]3$, unigrams plus bigrams.

Logistic head. Binary $P(y{=}1|\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b)$; K-class $P(y{=}k|\mathbf{x}) = e^{z_k} / \sum_j e^{z_j}$, $\mathbf{z} = W\mathbf{x} + \mathbf{b}$, trained under cross-entropy $-\log P(y|\mathbf{x})$.

Cross-encoder. Per option o_i, $s_i = \mathbf{w}_s^\top \mathbf{h}_i^{[\mathrm{CLS}]}$, $\mathbf{p} = \mathrm{softmax}(\mathbf{s})$, $\hat{y} = \arg\max_i p_i$; wide option sets train on gold plus seven distractors (sampled softmax over 8).

Calibration. Temperature $p_i = \mathrm{softmax}(\mathbf{z}/T)_i$ (shipped, one T per question); vector scaling $p_i = \mathrm{softmax}(\mathbf{z}/T + \mathbf{b})_i$ (fit, never shipped). Expected calibration error over B bins:

$$\mathrm{ECE} = \sum_{b=1}^{B} \frac{|B_b|}{n} \big|\mathrm{acc}(B_b) - \mathrm{conf}(B_b)\big|. \quad (4)$$

Unigram segmentation. For chunk c,

$$\mathbf{x}^* = \arg\max_{\text{seg}(c)} \sum_{\text{piece}\in\text{seg}} \text{score}(\text{piece}), \tag{5}$$

by Viterbi; unknown characters fall back to unk. Macro precision, recall, and F1 average the per-class values.

Abstention. Emit $\hat{y}$ iff $\max_i p_i \geq \alpha$ (`abstainAt`), else null—the fast System-One path answers, the slow path escalates.

Semantic routing (v2 prototype, parked). With train embeddings $\mathbf{e}(x)$ L2-normalized,

$$\mu_d = \frac{\sum_{x\in\text{train}_d} \mathbf{e}(x)}{\|\sum_{x\in\text{train}_d} \mathbf{e}(x)\|}, \quad \hat{d} = \arg\max_d \cos(\mathbf{e}(x), \mu_d), \tag{6}$$

kept iff $\cos \geq \tau$ (tuned on validation). Measured 0.289 absolute: parked.

Typed-decisions numbers use the public LocalLLaMA/typed-decisions test split, four workflows, same harness for nano and mini (argmax over options, macro scores). Competitor figures (OpenDecider, TypeSafe Jev, Laya) are vendor- or README-reported on the same named split, not re-measured by us; treat cross-row gaps as indicative, not audited. No competitor latency was measured; latency comparisons in the body refer only to our metered CPU figures. Wilson 95% intervals: mini 0.69 [0.62–0.75], nano 0.61 [0.54–0.67] (n=200); jailbreak 0.94 [0.91–0.96] (n=400); intent probes (n=10) span [0.60–0.98] and [0.72–1.00]—directional, not precise.

A plugin manifest declares identity (name, domain, version), the questions it answers, routing keywords and threshold, optional `languages` for shared domains, and supported modes. The router never inspects weights: routing is manifest-only, which is why a new skill ships as a repository, and why `bunkitsu` can resolve any plugin offline from a directory path.