

Adaptive Model Routing for Agentic Workloads

Knowing the Work Before the Agent Does It

Jacob Weiss

Head of AI, ProCap Financial, Inc. (NASDAQ: BRR)

jacob@cfosilvia.com

A Preprint · August 11, 2026

Abstract

Production chat assistants are increasingly powered by *agents*: a language model wrapped in a harness of tools and a control loop that the model iterates until the task is done. An instant factual lookup and a multi-minute deep-research report are produced by the *same* model and harness, and differ only in how many times the agent loops. Tracing the production turns of a frontier financial assistant, we find that the work is strongly bimodal in loop depth, and that this depth is predictable from the question text before the agent runs a single step. We formalize an agent as a model-harness pair, autonomous work as the agent composed with itself L times, and routing as choosing the cheapest model expected to clear a quality bar at the predicted depth; a sub-millisecond effort classifier (a linear softmax over hashed n -grams) then maps each turn to a model. Raw accuracy is modest (0.68 over a 0.58 base rate), but a router need only be right on the turns it acts on. Short-routing precision reaches 0.74 over half of all traffic and ~ 0.90 on its most confident slice, and the confidence is well calibrated, so the threshold can be read directly as a precision dial. Inference costs 0.34 ms at the median, four to five orders of magnitude below the model call it gates. At a balanced operating point roughly half of all *turns* can move to the smaller model. Because cost scales with loop depth and the routed turns are the shallow ones, the token-cost reduction is bounded above by $\sim 29\%$ at a $5\times$ price ratio, and it remains contingent on a quality check we do not yet measure: the classifier predicts frontier *effort*, not whether the smaller model answers acceptably. Finally, the question-only signal *saturates* near 0.72 ROC-AUC; the residual signal is in the *conversation state*. Adding leakage-safe session and user *effort-momentum* features lifts short-detector ROC-AUC by ~ 0.06 and accuracy by 3–4 points under a strict user-grouped split, rescuing terse follow-ups (“*yes, run it*”) that carry no lexical signal. Trained on the frontier agent’s own behavior, the router tracks the frontier as models and harnesses evolve rather than fighting it.

Keywords: agentic systems; model routing; cost-aware inference; text classification; LLM serving; latency.

1 Introduction

Every chat turn in our product currently runs on the most capable and most expensive frontier model available. This is a safe default and a poor economic one. Tracing production turns shows that the work is *bimodal*: a large share are shallow, near-instant lookups (“*what is today’s date?*”, “*what is the price of AAPL?*”),¹ while the remainder are deep, multi-tool analyses that consult skills, look up instructions, and execute code. Paying frontier-model token rates and frontier-model

¹Example queries throughout this paper are illustrative paraphrases, not verbatim user messages.

latency for the shallow tail is waste, and it slows down exactly the interactions where users most expect an immediate answer.

The central observation of this work is that the difference between a one-line answer and a deep-research report is not a difference of model or of harness. It is a difference of *depth*: how many times the agent iterates its loop before halting. If we can predict that depth from the question, before running it, we can spend compute proportional to the expected effort, routing shallow turns to a smaller, faster, cheaper model and reserving the frontier model for genuinely deep work.

Contributions.

1. A compact **formalism** for agentic work: an agent is a model–harness pair, autonomous work is the agent composed with itself L times, and both wall-clock time and token cost scale with L (Section 2).
2. A **routing objective** that selects the cheapest model expected to meet a quality bar at the predicted depth, and its reduction to a learned effort classifier when L is unobserved at decision time (Section 3).
3. A deployed **sub-millisecond classifier** (single-layer softmax over hashed n -grams) with a model card, a held-out evaluation, a served-model parity check, and a latency budget showing inference is never the bottleneck (Sections 4 and 7).
4. An **observability design** that logs every prediction next to the realized effort, turning production traffic into a continuous evaluation and retraining set (Section 8), and an **impact analysis** with a tunable precision/coverage operating curve (Section 9).
5. Evidence that the question-only signal is **lexical and saturating** (tuning, semantic embeddings, and a label-cut sweep all plateau near 0.72 ROC-AUC), and a **conversation-state extension** (leakage-safe session/user *effort-momentum* features) that breaks that ceiling (+0.06 ROC-AUC, +3–4 accuracy points under a user-grouped split), with the gain concentrated where the text is uninformative (Section 6).

Why now. This work is a direct product of tracing optimization for the latest frontier models. The routing target is *learned from how the current frontier agent actually behaves*, not from a hand-built notion of difficulty. When the model or harness changes, we re-trace and retrain; the classifier tracks the frontier rather than fighting it.

1.1 Related work

Cost-aware model selection for LLMs has two established shapes. *Cascades* run a cheap model first and escalate when its answer fails a check: FrugalGPT learns which sequences of model APIs to invoke per query [2], and AutoMix escalates based on few-shot self-verification of the small model’s own output [1]. Cascades pay for the small call (and the verification) on every escalated query. *Pre-call routers*, closer to this work, decide before any model generates: RouteLLM trains routers on human preference data to choose between a strong and a weak model [8]; Hybrid LLM predicts a per-query quality gap between the pair, with a tunable threshold much like our τ [3]; Shnitzer et al. [11] and FORC [9] learn per-model correctness or performance predictors from benchmark data; and RouterBench standardizes the evaluation of such routers [5].

In all of these, the routing signal is predicted *answer quality* (or human preference) on a stateless query, evaluated one prompt at a time.

A more recent line of work moves routing into the agentic setting, and is directly comparable to ours. TwinRouterBench evaluates routers at the level of individual *steps* within an agent trajectory rather than on one-shot prompts, pairing router-visible prefixes with execution-verified target tiers and measuring whether a cheaper substitution preserves downstream task success [13]. Agent-as-a-Router frames routing for agentic coding tasks and closes what it calls the router’s information deficit by accumulating execution-grounded feedback during deployment [14]. TRACE-Router observes that per-call independent decisions cost task-level coherence, and instead pins a task to one model via contextual bandits with delayed task-level rewards [10]. Our work shares their premise that per-call routing must be judged against whole-trajectory cost and outcome, and differs in two ways. First, these systems route *during* a trajectory, with the prefix in hand; we route once, *before* the first step, from the user message and conversation state alone, which is what makes sub-millisecond inference on the critical path possible. Second, they supervise on execution-verified downstream success, whereas our label is the frontier agent’s own realized effort, which is free at production scale but says nothing about whether the smaller model would have succeeded. That is a real limitation of our supervision, not a design choice, and it is the gap the quality gate of Section 3 is meant to close.

Two further differences separate this paper from the pre-call routers above. First, we route on realized *effort*, the loop depth L mined from production traces, rather than on estimated quality. In an agentic deployment, cost is not a per-call constant but scales with L (Section 2); the effort label tracks the deployed model–harness pair as both evolve. Second, where prior routers score the query text alone, we show the text-only signal saturates and that leakage-safe conversation-state features break that ceiling (Section 6). The serving posture, acting only on confident predictions and deferring to the frontier otherwise, is selective classification [4], and reading the threshold as a precision dial leans on the well-calibrated probabilities of logistic models [7]. The remaining ingredients are standard and cited where used: the hashing trick for features [12], model cards for documentation [6], and the tool-use loop that our agent formalism abstracts, popularized by ReAct [16].

2 A framework for agentic work

2.1 Agent = model + harness

Definition 1 (Agent). An *agent* is a pair $A = (M, H)$, where M is a language model and H is a *harness*: the tools, the control loop, and the context management that wrap M . A single step of the agent maps the current state s_t (the running context) to an action (text and/or tool calls) and a successor state:

$$A = (M, H), \quad s_{t+1} = f_A(s_t) = H[M](s_t). \quad (1)$$

Here $H[M]$ denotes the harness driving one decision of the model and applying its effects (tool results, appended context) to produce the next state. This is the standard tool-use loop of ReAct-style agents [16].

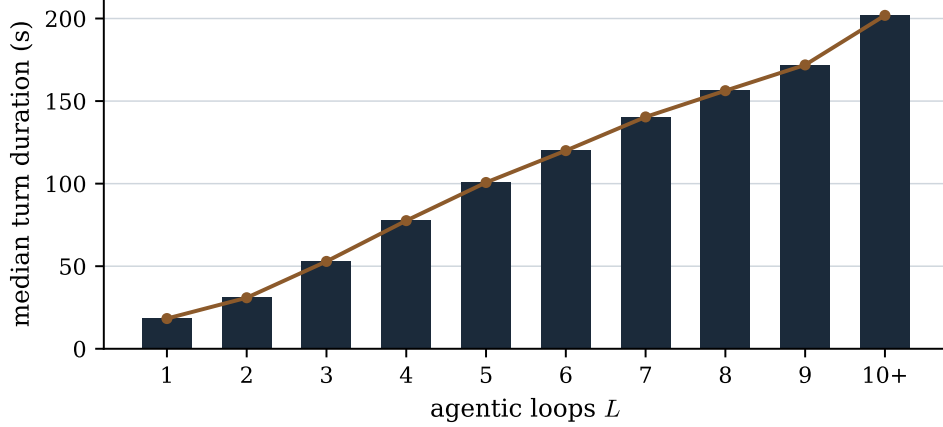


Figure 1. Median turn duration versus agentic loop depth over traced production turns. Duration grows approximately linearly, $T \approx L \bar{\tau}$, at $\sim 18\text{--}20\text{ s}$ per loop. Bars are per-bin medians; the line is the same series.

2.2 Work = the agent iterated over time

A task x seeds an initial state $s_0(x)$. The agent runs its loop until it halts; the number of steps $L(x)$ is the *effort*, or depth, of the autonomous work. Autonomous work is therefore the agent composed with itself L times:

$$\text{Work}_A(x) = (f_A \circ f_A \circ \dots \circ f_A)(s_0(x)) = f_A^{(L)}(s_0(x)), \quad L = L(x) \text{ loops.} \quad (2)$$

Effort classes are simply bands of L . A one-shot answer ($L \leq 1$), a few-step lookup (medium), a deep analysis (long), and a deep-research report ($L \approx 50\text{--}150$) are produced by the same M and H ; only the loop depth differs. A single classifier over the question text can therefore stand in for what is usually framed as a choice of product.

2.3 Time and cost both scale with depth

Each loop consumes wall-clock time and tokens. Writing $\tau_\ell(M)$ and $c_\ell(M)$ for the time and token cost of the ℓ -th loop under model M , the totals are sums over the realized loops, and are well approximated by the depth times a per-loop average:

$$T(x) = \sum_{\ell=1}^L \tau_\ell(M) \approx L \bar{\tau}(M), \quad C(x) = \sum_{\ell=1}^L c_\ell(M) \approx L \bar{c}(M). \quad (3)$$

Empirically, turn duration is near-linear in loop count at roughly 18–20 seconds per loop (Figure 1). Both time and cost therefore scale jointly with the depth L and with the chosen model’s per-loop speed and price, the two quantities a router controls.

3 Routing as prediction

Because cost and latency both scale with the pair (M, L) , the optimal model for a task is the *smallest one that still clears the quality bar* [2, 3]: there is no reason to pay frontier rates for a turn

a lighter model handles acceptably. Letting $\mathcal{M}$ be the available models, $q(M, x)$ a quality score, q_0 a threshold, and λ the price of latency relative to tokens, we choose

$$M^*(x) = \arg \min_{M \in \mathcal{M}} \mathbb{E}[C(M, L) + \lambda T(M, L)] \quad \text{s.t.} \quad q(M, x) \geq q_0. \quad (4)$$

The difficulty is that the depth L is *unknown before the turn runs*, and C, T in (4) both depend on it. The key empirical finding from tracing is that L is substantially predictable before the turn runs, largely from the question text. We therefore replace the unobserved L with a learned predictor of the effort *class* and route through a capability table $M(\cdot)$ that maps a class to a model:

$$\hat{y}(x) = \arg \max_k g_k(\phi(x)), \quad g(\phi) = \text{softmax}(W\phi + b), \quad M^*(x) = M(\hat{y}(x)). \quad (5)$$

Here $\phi(x)$ maps the question to a feature vector and g is a softmax classifier, equivalently a single-layer neural network. We train it on traces $\{(x_i, L_i)\}$ collected from the frontier agent itself, labeling each turn by its realized effort. The classifier thus learns the frontier model’s own effort distribution. It can be deepened later (contextual embeddings, an MLP head), but a linear model already separates the classes well and runs in microseconds.

A quality assumption we do not test. This reduction makes explicit an assumption that the objective (4) states but this paper does *not* verify: that a low-effort (**short**) turn is one the smaller model answers acceptably ($q(M_{\text{short}}, x) \geq q_0$). We treat “short effort $\Rightarrow$ small model suffices” as a working hypothesis for a quality gate (sample routed turns answered by the small model; shadow mode, Section 8) to confirm, not as a result established here. What we demonstrate is a predictor of frontier *effort*, not of answer quality.

Decision rule and the asymmetry knob. In the binary case the router compares the short-class probability against a threshold τ and falls back to the frontier model when it is not confident:

$$M^*(x) = \begin{cases} M_{\text{short}} & \text{if } g_{\text{short}}(\phi(x)) \geq \tau, \\ M_{\text{frontier}} & \text{otherwise.} \end{cases} \quad (6)$$

Because under-routing (sending hard work to the small model) is more costly than over-routing, τ is set above 0.5; it is the single knob that trades coverage for precision (Section 9).

Features. We use the hashing trick [12] to keep ϕ fixed-dimensional and allocation-free at serving time. Let $\mathcal{G}(x)$ be the multiset of word 1–2-grams and character 3–5-grams of the (truncated) question, h an FNV-1a hash, and $d = 2^{18}$. Then

$$\tilde{\phi}_j(x) = \sum_{g \in \mathcal{G}(x)} \mathbf{1}[h(g) \bmod d = j], \quad \phi(x) = \tilde{\phi}(x) / \|\tilde{\phi}(x)\|_2, \quad \phi(x) \in \mathbb{R}^d. \quad (7)$$

Feature hashing needs no vocabulary file, is robust to unseen tokens, and makes inference a single sparse dot product against the trained weights W .

Summary. The classifier does not predict the answer. It predicts how much work the answer will take, from the question alone, so that compute can be spent in proportion to that estimate.

Table 1. Model card for the text-only routing classifier (the v1 baseline). A later artifact extends it with the conversation-state features of Section 6 (Model C); the baseline is documented here because every text-only result in Sections 4–5 is measured on it.

Field	Value
Name / version	Text-only baseline; the later served artifact is Model C of Section 6
Task	Predict a chat turn’s effort class from its text; map to a model
Architecture	Multinomial logistic regression = single-layer softmax NN
Input	User message text (truncated to 512 chars)
Features $\phi(x)$	Hashed word (1–2g) + char (3–5g) n -grams, FNV-1a $\rightarrow 2^{18}$ dims, L_2 -normalized
Output	Class (short not_short) + per-class probability + suggested model
Routing map	short $\rightarrow$ the smaller in-family model; default / not_short $\rightarrow$ the frontier model (env-tunable)
Training data	Unique production turns from a recent window of traffic; stratified 80/20 train/held-out split (absolute volumes withheld; see the data-availability statement)
Label	Realized effort: short = $\text{loop_count} \leq 5$, else not_short
Held-out metrics	accuracy 0.675 $\cdot$ macro-F1 0.653 $\cdot$ short-detector PR-AUC 0.761 (clean split)
Served-model check	Serving-parity check: 0.697 on a re-pulled split that overlaps training (confirms the served path reproduces the offline pipeline; not a generalization estimate)
Artifact	Sparse weights, 241,657 nonzero rows (~ 6.5 MB JSON), CPU-only
Inference	Sparse dot product; ~ 0.34 ms p50 server-side
Intended use	Pre-call model selection for the chat experience
Out of scope	Deep-research pipeline (always frontier); safety / abuse decisions

4 The routing model

Table 1 summarizes the deployed baseline as a model card [6]. The artifact is a single JSON file baked into the serving image and loaded into memory; the prediction endpoint is pure (no database writes on the hot path). The label is the *realized* loop count bucketed at $L \leq 5$, chosen after inspecting the distribution of tool calls and loop counts on the training corpus (Section 4.1).

4.1 Training data

We trace turns from the production trace store, label each by its realized effort, and deduplicate by message. The *training* corpus is a recent window of that traffic rather than the full trace history: the label is realized behavior, so turns produced under an older harness describe an effort distribution the deployed agent no longer has, and training across both would fit an obsolete target. (Absolute corpus and traffic volumes are withheld throughout; see the data-availability statement. All results are reported as shares, rates, and per-example metrics.)

We label these turns with a single loop-count cut. Figure 2 shows the loop distribution: mass is centered at $L = 3\text{--}5$ (median 5), so it is not dominated by one-shot answers ($L = 1$ is only 11%). A cut at $L \leq 5$ gives a balanced, well-separated split (58.0% **short** / 42.0% **not_short**) with real volume on both sides of the routing decision; stricter cuts (e.g. $L \leq 1$) folded ordinary two- and three-step answers into the deep class. Each turn is **short** if $L \leq 5$ and **not_short** otherwise.

Length is not the signal. Median message length barely differs between light and heavy turns (77

vs 87 characters; 15 vs 17 tokens), and the lighter turns actually have the *longer* p90 (434 vs 361 characters), so a length threshold or keyword rule is useless to counterproductive, which is why the classifier reads hashed n -grams rather than surface statistics.

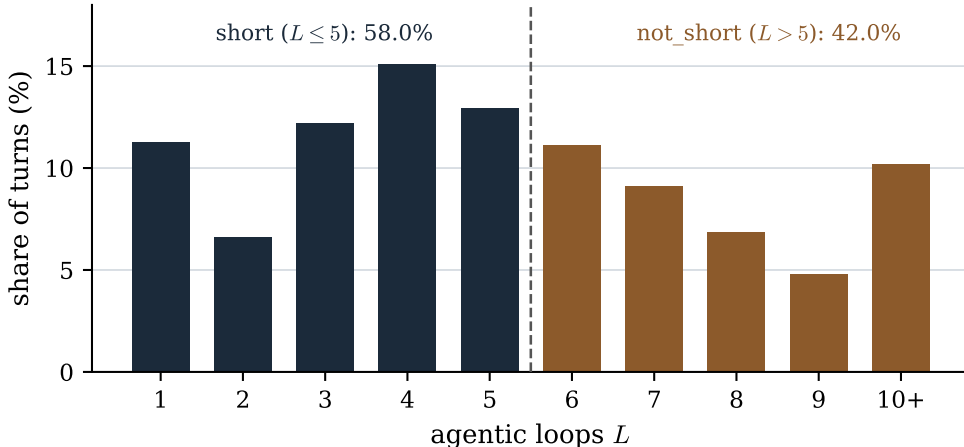


Figure 2. Loop-count distribution over the training corpus (share of turns). The $L \leq 5$ cut (dashed) splits the corpus 58.0% short / 42.0% not_short, with substantial volume on both sides.

4.2 Held-out evaluation

On the held-out split the deployed model attains accuracy 0.675, macro-F1 0.653, and short-detector PR-AUC 0.761. We separately re-run the *served* artifact end-to-end as a serving-parity check, asking whether the deployed featurizer and weights compute the expected function. On a freshly pulled holdout it scores 0.697, but this is *not* an independent generalization estimate: the re-pull re-splits by row position, so with a changed corpus size and order it evaluates largely on turns that were in training. We report it only to confirm the served path reproduces the offline pipeline, and keep the clean 0.675 as the generalization number. Figure 3 is the confusion matrix at the default decision point (argmax, $\tau = 0.5$).

The error structure is what matters for routing. Short turns are caught well (recall 0.80). The expensive cell is the **under-route** (outlined): true not_short turns predicted short, that is, deep work sent to the small model. At the default $\tau = 0.5$ these are 49.6% of all hard turns, and 31% of everything routed to the small model is an under-route (one minus precision). Because under-routing is the costly error (Section 3), the router raises τ above 0.5, trading coverage for precision. The full curve is Figure 11 (Section 9).

Confident precision, not accuracy, is the product. A router need not be right on every turn, only on the ones it chooses to move, which makes the relevant measure a selective one [4]: how reliable is the short call when the model is confident? Sweeping the confidence threshold τ (Figure 11), short-routing precision rises monotonically from 0.69 at $\tau=0.5$ to 0.74 while still covering half of all traffic ($\tau=0.6$), 0.79 at $\tau=0.7$, and ~ 0.90 on the most confident slice, staying above the 58% base rate throughout. The errors concentrate in the low-confidence region, which is exactly where the router abstains and defers to the frontier model, so the price of caution is paid in coverage rather than in mistakes.

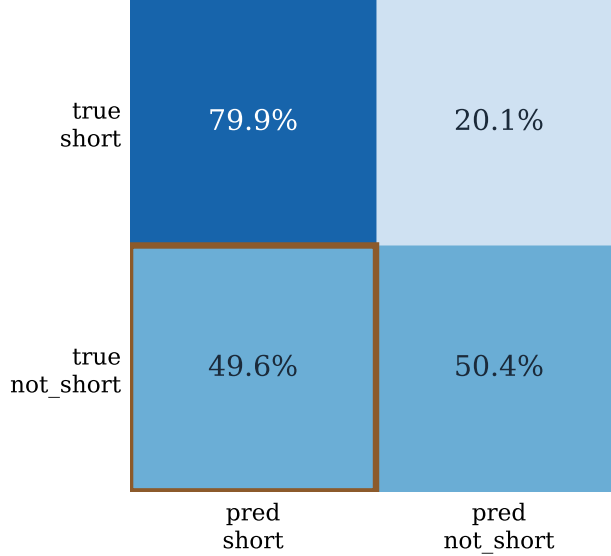


Figure 3. Held-out confusion matrix, row-normalized, at $\tau = 0.5$. The outlined cell is the costly *under-route*: deep (not_short) turns sent to the small model.

Calibration. Thresholding on g_{short} is only meaningful if the probability means what it says, so we check it directly. Figure 4 bins the held-out turns by predicted $P(\text{short})$ and plots the observed short rate per bin: the curve tracks the diagonal closely across the full range (e.g. the 0.9–0.95 bin realizes 0.92). No post-hoc calibration layer is needed, which is consistent with the well-calibrated probabilities logistic models are known to produce [7], and it is what licenses reading τ as a precision dial rather than an arbitrary score cutoff.

4.3 Limitations and risks

- **The label is realized behavior, not intrinsic difficulty.** Effort is what the frontier agent *did*; the same question can change class if the agent’s tool-use policy changes. A single harness change can invert much of the label distribution, so the target is non-stationary and the router can silently rot between retrains; shadow-mode monitoring (Section 8), not retrain cadence alone, is the real mitigation.
- **Error costs are asymmetric.** Under-routing is worse than over-routing. The threshold τ in (6) is tuned for precision, and low-confidence turns fall back to the frontier model.
- **Distribution coverage.** The model is trained on finance-assistant traffic; out-of-domain text routes conservatively to the frontier model by construction of (6).
- **Possible split leakage.** Rows are deduplicated keep-first on lowercased text and split randomly, not grouped. Heavily templated finance queries (the same question with a different ticker) share n -grams and can land on both sides of the split, so the held-out numbers may be modestly optimistic; the group-held-out re-evaluation in Section 6 (whole conversations and users held out) scores the text-only model comparably, so the leakage appears modest in practice.
- **No temporal holdout.** All splits here are random or grouped by session/user, not by time, yet the target drifts whenever the harness changes. The deployment-relevant generalization estimate

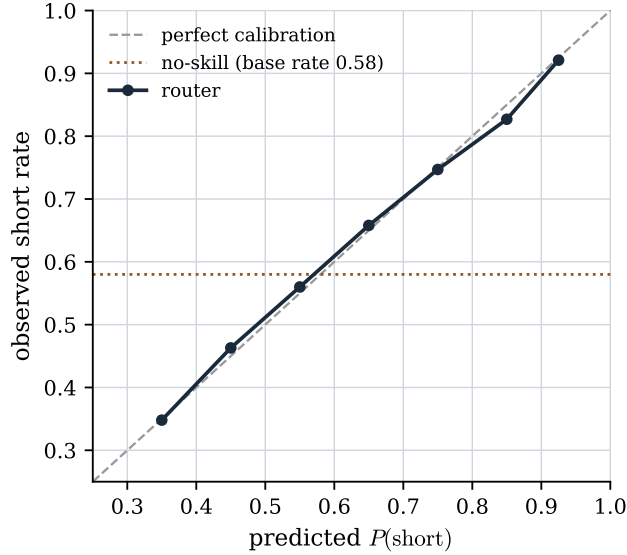


Figure 4. Reliability diagram for the short-detector probability on the held-out split: observed short rate versus predicted $P(\text{short})$, binned. The router tracks the diagonal (perfect calibration) across the range; the dotted line is the no-skill base rate.

is a forward-in-time split (train on the earlier weeks of the window, test on the latest), which we have not run; the reported numbers should be read as within-window performance, with shadow-mode online accuracy (Section 8) as the true forward test.

5 The text-only ceiling

The classifier of Section 4 reads only the user’s message. Pushing that single input as hard as it goes, three independent attempts to raise accuracy all plateau in the same place.

- **Hyperparameter tuning.** A cross-validated grid over the inverse regularization C , class weighting, and the word/character n -gram ranges moves short-detector PR-AUC by less than the cross-fold standard deviation (± 0.001). The only featurizer change that helps at all, adding word trigrams, buys ~ 0.005 PR-AUC for a 10% larger artifact and does not clear a no-regression bar.
- **Semantic embeddings.** Replacing the hashed n -grams with a frozen sentence encoder (bge-small-en-v1.5 [15], 384-dim, CLS-pooled) plus a logistic head is *worse* than n -grams (PR-AUC 0.741 vs 0.763); an MLP head only ties. A general-purpose encoder is trained to map paraphrases together, which compresses away the exact lexical cues that signal effort. Two questions can be semantically near yet opposite in effort. “*What is my portfolio worth*” is a one-shot lookup and “*should I rebalance my portfolio*” is a multi-step analysis, but the embedding makes them look alike. The signal is *lexical*, and a linear model over n -grams already captures it.
- **Label cut.** Sweeping the threshold k (with **short** defined as loop count $\leq k$), the base-rate-invariant separability (ROC-AUC) is highest at $k = 1$ and decays as the cut climbs into the lexically fuzzier 2–5-loop band (Figure 5). Under the product constraint that the small model

must absorb work up to ~ 5 loops ($k \geq 5$), the deployed $k = 5$ is already the most separable admissible cut; the rising *accuracy* at larger k is base-rate inflation (at $k = 10$, 93% of turns are short, so a constant predictor scores 0.93).

When more features, a more expressive model, and a different label all converge on the same number, the bottleneck is *information*, not model capacity: the question text alone does not separate effort beyond ~ 0.72 ROC-AUC.

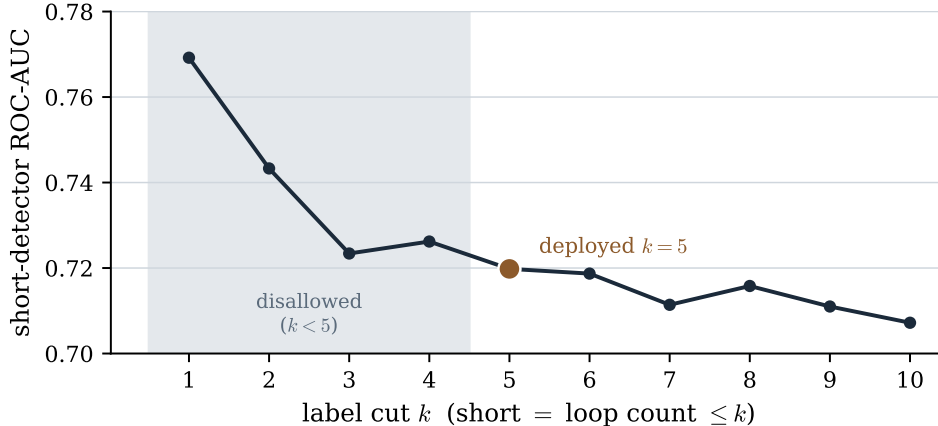


Figure 5. Short-detector ROC-AUC (base-rate-invariant) by label cut k on the refreshed held-out corpus. Separability peaks at the trivially-short ($k = 1$) boundary and erodes upward; within the admissible region ($k \geq 5$, unshaded) the deployed $k = 5$ is already the best cut.

6 Effort momentum: conversation state as a feature

If the message is saturated, the missing signal must lie elsewhere, and the framework of Section 2 says where. The seed state $s_0(x)$ is not the bare question but the *running conversation*, so the depth $L(x)$ is a property of the session, not of the turn text in isolation. Empirically, effort is strongly **autocorrelated**: a turn in the middle of a deep analysis tends to stay deep, and a lookup is followed by lookups. The agent’s recent realized loop depth is therefore a feature for the next turn, which we call **effort momentum**.

Features (leakage-safe). To each turn we append, computed from *strictly prior* turns only: the within-session prior loop counts (the immediately previous count, the running mean and max, the turn index, and whether the turn is the session opener); the user’s recent loop counts across sessions (previous and mean); and three cheap surface features of the message (token count, character count, and a *terse* flag for ≤ 3 tokens). The current turn’s loop count, which is the label, is never an input, and every prior-turn feature is available at decision time because those turns have already completed.

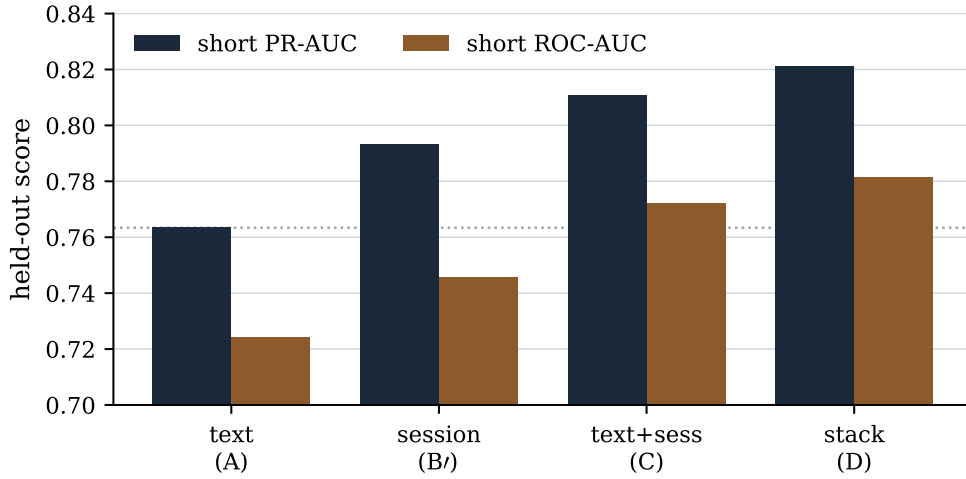
Evaluation (grouped, no dedup). We re-trace the same traffic window carrying a session id and a user id and keep the per-session turn order (no message-level dedup); 81.5% of the turns are follow-ups. Because the momentum features are computed within a session, the held-out split must keep whole conversations together; we split by *group*, not by row, which directly addresses

Table 2. Effort-momentum features versus text alone, on the session-grouped held-out split (short base rate 0.57). All rows share the split; read the lift over the text-only baseline (A).

	Model (inputs $\rightarrow$ head)	Acc.	Macro-F1	PR-AUC	ROC-AUC
A	text n -grams $\rightarrow$ linear	0.676	0.656	0.763	0.724
B'	session features $\rightarrow$ GBT	0.685	0.673	0.793	0.746
C	text + session $\rightarrow$ linear	0.706	0.692	0.811	0.772
D	text score + session $\rightarrow$ GBT	0.713	0.702	0.821	0.781

the leakage caveat of Section 4. Absolute numbers shift slightly from Section 4.2 (different dedup and split), so the decision-relevant quantity is the lift over the text-only model on the *same* split.

Results. Table 2 and Figures 6 and 7 report four models on the session-grouped split. Session features alone (B') already rival text; combined (C, D) they dominate it. The best model, a gradient-boosted stack over the text model's score and the momentum features (D), lifts short-detector ROC-AUC from 0.724 to 0.781 (+0.057) and accuracy from 0.676 to 0.713 (+3.8 points), an order of magnitude more than any text-only change of Section 5. The linear combination (C) keeps most of the gain in a single dot product. The two signals are complementary: the text carries the session opener, momentum carries the follow-ups.

**Figure 6.** Short-detector PR-AUC and ROC-AUC for text only (A), session features alone (B'), and the combined models (C linear, D gradient-boosted stack), on the session-grouped held-out split. Dotted line: the text-only PR-AUC.

A conservative confirmation. The user-history features could in principle let a model recognize a user seen in training. To rule that out we re-split by *user*, holding every user's turns entirely in or out, the strictest leakage-free setup. The lift is undiminished: D reaches accuracy 0.723, ROC-AUC 0.791, and PR-AUC 0.843, a +0.060 ROC-AUC and +3.3-point accuracy gain over text-only (0.690 / 0.730). The result is driven by the within-session features, which are clean under any grouping.

Where the lift comes from, and the follow-up fix. Figure 8 breaks accuracy down by turn type (user-grouped). The gain is broad. It is largest on *session openers* (+4.6 points): there is no

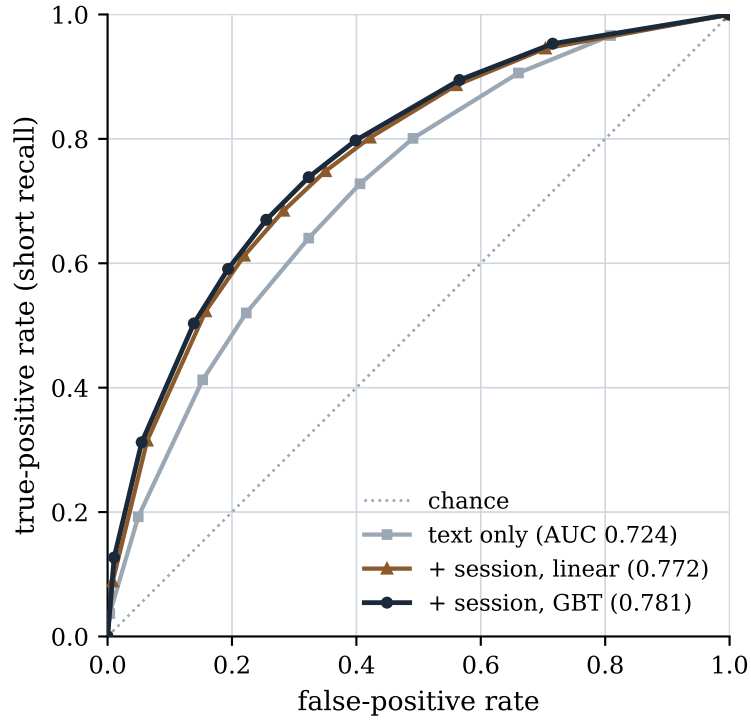


Figure 7. ROC curves on the session-grouped held-out split: text only (A), text + conversation-state features (linear, C), and the gradient-boosted stack (D). Effort momentum shifts the whole curve toward the top-left, lifting short-detector ROC-AUC from 0.724 to 0.781. Each threshold τ is one point on a curve; the dotted diagonal is chance.

momentum yet, but knowing that a turn is the opening ask, which skews deep (31% **short**), together with the user’s cross-session history corrects the text model’s bias. For the motivating case, **terse follow-ups** such as “yes, run it”, which carry no lexical effort signal, improve from 0.727 to 0.751: the model now reads what the session was already doing. (Accuracy understates the win on that slice, which is 72% **short**; the real value is catching the minority that are secretly deep, giving fewer dangerous under-routes.)

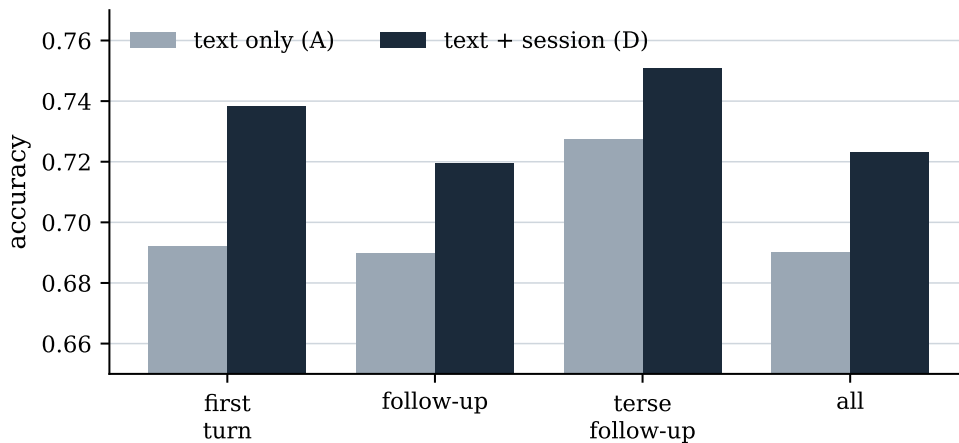


Figure 8. Held-out accuracy by turn type (user-grouped split): text only (A) versus text + session (D). Every turn type improves; the largest gains are on session openers, and the motivating terse follow-ups are rescued.

Operating curve. The momentum model also dominates the routing trade-off (Figure 9): at a matched $\sim 50\%$ coverage, short-routing precision rises from 0.73 (text only) to 0.78, cutting the costly under-route rate from 27% to 22%, and the curve sits above the text-only curve at every operating point.

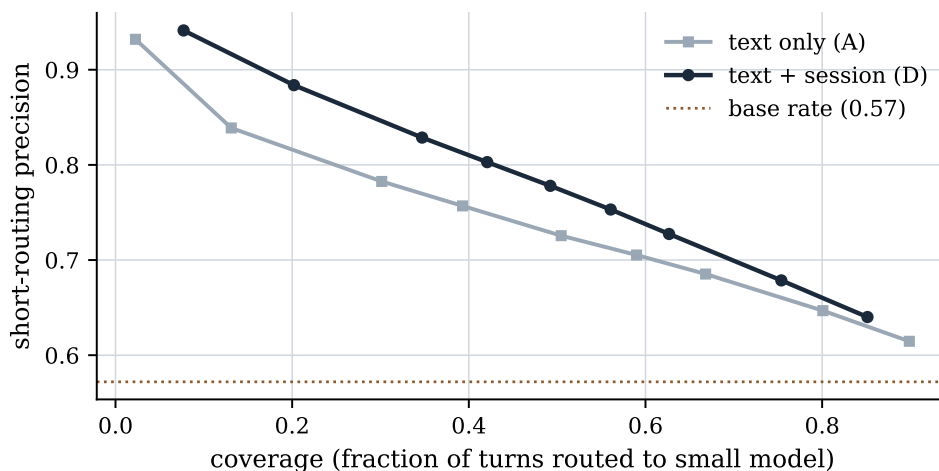


Figure 9. Short-routing precision versus coverage (sweeping τ): text only (A) versus text + session (D), session-grouped holdout. The momentum model dominates throughout; the dotted line is the base rate.

Table 3. Latency budget for the routing path (compute is never the bottleneck). The compute row is a microbenchmark of the baseline artifact; the round-trip row is from a single probing session (indicative), and the frontier-call reference reuses the median single-loop turn duration as a proxy.

Layer	p50	p90	p99
Classifier compute (inference)	0.34 ms	0.71 ms	1.21 ms
Endpoint round-trip (network-bound)	75 ms	84 ms	213 ms
A single frontier model call (reference)	~18s	—	—

Serving implication. Momentum requires no change to the classifier’s architecture. The predictor remains a pure, stateless function of its inputs; the prior-turn loop counts it needs are already available to whatever holds the conversation, so they can be supplied alongside the message text and the features derived at prediction time. Model C is a drop-in extension of the same sparse dot product, and both C and D consume identical inputs, so moving between them does not change what a caller must provide.

7 Latency

The classifier is designed to be effectively free relative to a model call. Table 3 reports measurements on the baseline artifact. Inference is a sparse dot product over the hashed features and costs 0.34 ms at the median; the end-to-end round trip is dominated entirely by the network, and a single frontier model call is four to five orders of magnitude slower still.

Under a live concurrent load test against the deployed endpoint (Figure 10), the router sustains 358 requests per second with *zero errors*, and even at $64\times$ concurrency the tail stays sub-second. The median round trip rises only modestly under load, from ~ 76 ms unloaded to ~ 120 ms at peak, because the cost is the network and request handling, not the model: the classifier itself is a sub-millisecond dot product, and the session-context features of Section 6 add no measurable serving cost. The network round trip is thus ~ 2 orders of magnitude cheaper than the ~ 18 s frontier call it gates, and the classifier compute itself 4–5 orders. Routing is never the bottleneck, which is what makes spending compute proportional to predicted effort essentially free.

The implication for optimization is clear: the next round targets the network path rather than the model. Any such change must leave the artifact unchanged so that routing decisions remain consistent across the optimization.

8 Logging and observability

The router serves a pure prediction endpoint; the application is responsible for logging. Every decision is written to the analytics store on the same row as the turn’s *realized* effort, closing a measurement loop with no separate evaluation job:

- **Logged per turn:** predicted class, predicted model, confidence, router mode, and the realized `loop_count` and tool usage, together in one record.
- **Online accuracy** is then predicted-versus-realized, computable directly in SQL.
- **Staged rollout:** off $\rightarrow$ shadow (log only, zero risk) $\rightarrow$ live. Shadow mode proves accuracy on live traffic before any response is changed.

Ultra-low-latency model routing for adaptive workloads

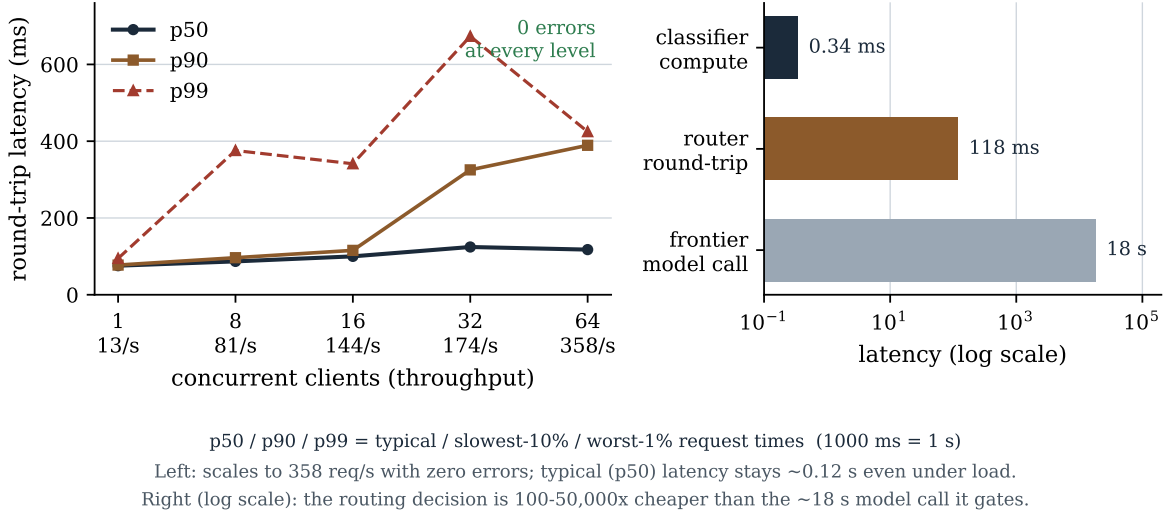


Figure 10. Ultra-low-latency routing. *Left:* round-trip latency to the deployed prediction endpoint under a concurrent load test; throughput scales to 358 req/s with zero errors and a near-flat median. *Right:* the end-to-end latency budget on a log scale. Classifier compute (0.34 ms) and the network round trip (~120 ms under load) are dwarfed by the ~18 s frontier model call the router gates.

- **Continuous retraining:** the same logs are the next training set, so the router tracks the frontier model as it evolves.

9 Expected impact

Routing is a tunable knob. The threshold τ of (6) trades *coverage* (how much traffic moves to the smaller model) against *precision* (how often that is the right call). Figure 11 shows the held-out operating curve; the smaller model is roughly 5 $\times$ cheaper per token and faster per loop. The conversation-state model of Section 6 raises this entire curve (Figure 9).

Savings must be weighted by cost, not turn count. Equation (3) gives $C \approx L \bar{c}$, and the router deliberately sends the *low-L* turns to the cheap model, so a turn-count estimate overstates the token saving. From the training-corpus loop distribution, short ($L \leq 5$) turns are 58% of turns but only ~36% of all agentic loops. Routing *every* true-short turn therefore caps the saving at

$$\text{token-cost reduction} \lesssim (\text{loop share}) \times (1 - \rho) \approx 0.36 \times \left(1 - \frac{1}{5}\right) \approx 29\%,$$

where $\rho \approx 1/5$ is the small-to-frontier per-token price ratio. The realized figure is lower: the conservative operating points actually used route fewer turns, precision is imperfect (under-routes are deep, high- L turns), and later loops carry more accumulated context, so deep turns dominate spend even more than their loop count implies. The bound also assumes the smaller model matches the frontier’s loop count on routed turns; the labels measure the work *the frontier* did, and a turn the frontier closes in one loop may take the smaller model several, eroding the effective price ratio ρ . We therefore report ~29% as an *upper bound*, not a point estimate; a precise per-operating-point

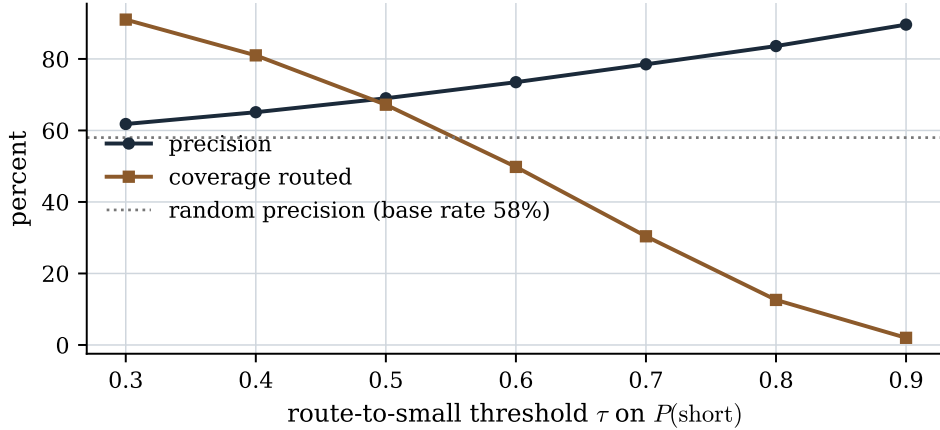


Figure 11. Measured held-out operating points (the deployed model’s threshold sweep). Lowering τ routes more traffic to the small model at lower precision; raising it is conservative. The dotted line is the base-rate precision (58%) a random router attains at any coverage; the model sits above it throughout. The router picks a point on this curve.

Table 4. Measured operating points (held-out): turn-level coverage routed to the small model. Token-cost savings are bounded above by $\sim 29\%$ (see text), and are *not* equal to the turn coverage.

Operating point	Turns routed to small model	Posture
$\tau = 0.50$	$\sim 67\%$	Aggressive
$\tau = 0.60$	$\sim 50\%$	Balanced
$\tau = 0.70$	$\sim 30\%$	Conservative

figure needs per-example loop weighting, which we defer to the productionization eval (with the quality gate above). Table 4 lists the measured turn coverage at three thresholds.

Net savings also depend on escalation and retry rates and on the (unmeasured) quality of the small model on routed turns; see Section 4.2. Latency improves in parallel: shallow turns finish on a faster model, often in fewer loops.

Bottom line. Spend compute proportional to predicted effort. The frontier model remains the default for hard work and easy work gets fast, cheap answers, with shadow mode verifying that the decision is right before any response changes.

10 Next steps

- **Online validation of effort momentum.** The conversation-state model of Section 6 is evaluated offline only. The next step is to confirm that its measured lift holds on live traffic before it informs any routing decision.
- **Quality gate.** Sample routed turns, assess whether the smaller model’s answers meet the product’s bar, and set τ from that result. This is the assumption of Section 3 that the present work does not test.
- **Forward-in-time evaluation.** Every split reported here is random or grouped by session/user.

A temporal split is the deployment-relevant generalization estimate, and the drift in the target is why it matters.

- **Retraining.** The target is non-stationary (Section 4.1), so the classifier has to be retrained as the underlying model and harness evolve.

Data availability and privacy

The training data derives from private customer conversations traced in production and is not released; neither are the model artifact or the internal experiment code. This paper instead shares the framework, the measurement methodology, and the findings, so that others can apply the same approach to their own traces. Traces were used in accordance with the product’s terms of service and privacy policy. Only aggregate statistics appear in this paper: no message content, user identifiers, or per-user data are published, and the example queries shown are illustrative paraphrases rather than verbatim messages. Absolute traffic volumes (turn, session, and user counts) are likewise withheld as business-confidential and reported as shares and rates instead; all evaluation splits are large enough that the sampling error of the reported rates is negligible at the precision shown.

References

- [1] P. Aggarwal, A. Madaan, A. Anand, S. P. Potharaju, S. Mishra, P. Zhou, A. Gupta, D. Rajagopal, K. Kappaganthu, Y. Yang, S. Upadhyay, M. Faruqi, and Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. [arXiv:2310.12963](#).
- [2] L. Chen, M. Zaharia, and J. Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024. [arXiv:2305.05176](#).
- [3] D. Ding, A. Mallick, C. Wang, R. Sim, S. Mukherjee, V. Rühle, L. V. S. Lakshmanan, and A. H. Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *International Conference on Learning Representations (ICLR)*, 2024. [arXiv:2404.14618](#).
- [4] Y. Geifman and R. El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 4878–4887, 2017. [arXiv:1705.08500](#).
- [5] Q. J. Hu, J. Bieker, X. Li, N. Jiang, B. Keigwin, G. Ranganath, K. Keutzer, and S. K. Upadhyay. RouterBench: A benchmark for multi-LLM routing systems. [arXiv:2403.12031](#), 2024.
- [6] M. Mitchell, S. Wu, A. Zaldivar, P. Barnes, L. Vasserman, B. Hutchinson, E. Spitzer, I. D. Raji, and T. Gebru. Model cards for model reporting. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT*)*, 2019. [arXiv:1810.03993](#).
- [7] A. Niculescu-Mizil and R. Caruana. Predicting good probabilities with supervised learning. In *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, pages 625–632, 2005.
- [8] I. Ong, A. Almahairi, V. Wu, W.-L. Chiang, T. Wu, J. E. Gonzalez, M. W. Kadous, and I. Stoica. RouteLLM: Learning to route LLMs from preference data. In *International Conference on Learning Representations (ICLR)*, 2025. [arXiv:2406.18665](#).
- [9] M. Šakota, M. Peyrard, and R. West. Fly-swat or cannon? Cost-effective language model choice via meta-modeling. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining (WSDM)*, 2024. [arXiv:2308.06077](#).

- [10] R. Raj, S. Kundu, S. Banerjee, D. Joshi, I. Vohra, and T. Krishna. TRACE-ROUTER: Task-consistent and adaptive online routing for agentic AI. [arXiv:2607.22465](#), 2026.
- [11] T. Shnitzer, A. Ou, M. Silva, K. Soule, Y. Sun, J. Solomon, N. Thompson, and M. Yurochkin. Large language model routing with benchmark datasets. [arXiv:2309.15789](#), 2023.
- [12] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th International Conference on Machine Learning (ICML)*, pages 1113–1120, 2009. [arXiv:0902.2206](#).
- [13] P. Yang, W. Chen, T. Yang, P. Feng, J. Xing, W. Guo, Y. Yao, Y. Han, H. Li, X. Wang, Z. Wang, J. Xiao, A. Yang, L. Tian, L. Ai, E. Yang, and T. Shi. TwinRouterBench: Fast static and live dynamic evaluation for realistic agentic LLM routing. [arXiv:2605.18859](#), 2026.
- [14] P. Zhou, Z. Tang, Y. Ma, J. Tang, Y. Han, Z. Wan, F. Meng, W. Wang, B. Zhuang, W. Zhao, and Y. You. Agent-as-a-Router: Agentic model routing for coding tasks. [arXiv:2606.22902](#), 2026.
- [15] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie. C-Pack: Packed resources for general Chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024. [arXiv:2309.07597](#).
- [16] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. [arXiv:2210.03629](#).