WHEN TO CONSTRAIN, GUIDE, OR DEFER: ADAPTING FROZEN LANGUAGE MODELS TO PRIVATE INVENTORIES

Anonymous authors
Paper under double-blind review

ABSTRACT

In this paper, we explore how to adapt a frozen, served language model to a private, user-specific inventory, such as a model hub, a tool catalog, or a database schema, without modifying its weights. Focusing on the sampling space, we study two complementary mechanisms for inference-time adaptation: constrained decoding and a small expert trained on user-side usage logs. We analyze when each is beneficial, at what granularity it should be applied, and how they should be composed. In our experiments, constrained decoding yields gains only where the base model already possesses partial knowledge of the relevant names (+6 on the primary model where it does, down to -3 where it does not; the boundary holds across the 46 host–catalog pairs we can probe), while the neural expert helps only within its training scope (+21 on its catalog, down to -32 off it). We further find that the signal for deciding when an expert should act is weak in thresholded token-level statistics, only partly recovered by a learned per-token head, and present in a representation of the full request. We propose a prefill-based router that reads the expert's own prefill state to either apply adaptations or default to the base model. The same request-level verdict can activate both mechanisms at once: when both apply, the mask adds its guarantee that no invented name is emitted. We observe that the routed, unmasked coupling adds 16 points of accuracy on its target dataset and returns the base model exactly on unrelated math questions.

1 INTRODUCTION

In real-world deployments, language models face a dual mandate: they must act as reliable experts on specific, proprietary inventories (private function catalogs (Patil et al., 2025), internal model hubs (Patil et al., 2024), confidential database schemas (Li et al., 2023a; Feng et al., 2025)) while preserving their broad, generalist capabilities on every other query (Ling et al., 2025; Qin et al., 2024; Yao et al., 2024). Standard deployments make this difficult. Foundation models (open-source or proprietary) are typically served as frozen endpoints (OpenAI, 2023; Bommasani et al., 2021). Furthermore, the behavior of proprietary APIs can change unpredictably (Chen et al., 2024a). Enterprise governance can also prohibit uploading private data for provider-side fine-tuning (European Data Protection Board, 2024; Menlo Ventures, 2024). Because requests arrive as an unpredictable, mixed stream, user-side adaptation must supply private-inventory knowledge while determining when to leave the host unchanged. With weights and hidden states out of reach, there remain two primary options: inputs and sampling. Practice has largely turned to the input space via prompt engineering, which inflates context costs and still cannot guarantee valid outputs (Edge et al., 2024; Gao et al., 2023; Tian et al., 2024). This leaves the sampling space as another axis of control. Self-hosted models expose their next-token distribution directly (Willard & Louf, 2023; Dong et al., 2025), while closed models increasingly offer similar control via structured outputs (OpenAI, 2024).

Within the sampler lever, two families of work touch the problem. (1) *Constrained decoding* (grammar- or schema-derived masks): it guarantees that an emitted identifier exists (Scholak et al., 2021; De Cao et al., 2021; Zhang et al., 2024c; Willard & Louf, 2023; Dong et al., 2025; Ugare et al., 2025), and is known to trade validity against accuracy (Park et al., 2024; Tam et al., 2024; Li et al., 2026; Lee et al., 2026; Maninger et al., 2026). What it does not say is *where* it pays; it cannot add what the host does not know (Wen et al., 2026; El Hamdani et al., 2025). (2) *Neural guidance*: a second model reshapes

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

the host's next-token distribution, whether a tuned small expert blended at the logits (Liu et al., 2021; 2024; Khandelwal et al., 2020; Zhang et al., 2024b), a learned per-token deferral (Shen et al., 2024), or a value/reward model steering toward an attribute (Yang & Klein, 2021; Deng & Raffel, 2023; Mudgal et al., 2024; Han et al., 2024; Xu et al., 2025); all act token by token. None of them asks, per request, whether the guidance should apply at all. These two families move in opposite ways: the first guarantees legality but cannot add knowledge; the second adds knowledge but has no request-level off-switch.

To close this gap, we adapt request-level routing (Chen et al., 2024b; Ong et al., 2025). Thus, we propose a prefill-based router that reads the expert's internal state to decide, per request, whether the expert and the mask apply. We evaluate our approach on closed-inventory tasks where legality and correctness can be scored exactly: three API catalogs of varying familiarity, function calling with controlled candidate sets, text-to-SQL, and text-to-Cypher, alongside GSM8K as an out-of-scope control (Patil et al., 2024; 2025; Li et al., 2023a; Feng et al., 2025). Our contributions: **(1)** We analyze when constrained decoding benefits frozen models on private legal sets. We introduce an offline probe-set measurement that predicts this gain, validated across 16 open models and 3 catalogs. **(2)** We diagnose expert guidance, showing small private experts improve in-domain accuracy but degrade out-of-domain. We find the routing signal weak in thresholded per-step statistics, partly recovered by a learned per-token head, and present in the full-request representation. **(3)** We propose an adaptive strategy where a request-level router leverages the expert's prefill state to dynamically activate both the expert and constraint mask. On closed APIs, the routing policy and legal sets transfer successfully even when the expert does not. We conclude that on the user's side of a served model, legality belongs to the sampler, knowledge to a user-owned expert, and applicability to the request.

2 Related work

The challenge of grounded generation on unseen inventories has been approached from multiple paradigms: in-context retrieval for tool selection (Patil et al., 2024; 2025; Qin et al., 2024), vocabulary encoding via training (Wang et al., 2025), schema-linking and prompt serialization for text-to-SQL (Wang et al., 2020; Li et al., 2023a; Safdarian et al., 2026; Chang et al., 2023), closed-entity and knowledge-graph grounding (Feng et al., 2025; Gu et al., 2021; Gu & Su, 2022; Edge et al., 2024; De Cao et al., 2021), and broad adaptation via fine-tuning (Hu et al., 2022), activation steering (Turner et al., 2025), or reflection training (Zhang et al., 2024a; Asai et al., 2024). In all these approaches, the inventory enters through the prompt, custom architectures, or model weights. When the serving interface exposes neither weights nor internal states, the sampling space is the alternative to the prompt.

2.1 Constrained decoding

Constrained decoding restricts the next-token distribution to a formal language, a mechanism now efficiently supported by modern inference engines via token-level masking and by proprietary APIs as structured outputs (Scholak et al., 2021; De Cao et al., 2021; Zhang et al., 2024c; Agrawal et al., 2023; Willard & Louf, 2023; Dong et al., 2025; Ugare et al., 2025; Beurer-Kellner et al., 2024; Park et al., 2025; Geng et al., 2023; OpenAI, 2024). However, enforcing strict legality carries two known costs: (1) Masking is not conditioning: lacking lookahead, it forces the model into legal trajectories it would not naturally generate, causing strict restriction alone to degrade reasoning (Park et al., 2024; Banerjee et al., 2025; Tam et al., 2024; Lee et al., 2026; Li et al., 2026). (2) Validity does not guarantee correctness (Maninger et al., 2026; Wen et al., 2026; El Hamdani et al., 2025). While empirical gains are task-specific and costs are systematically documented (Lee et al., 2026), prior work lacks a framework to determine when constraints are beneficial across varying models and inventories, or how to predict it from an offline probe set.

2.2 Neural guidance

A second paradigm alters the host's next-token distribution through auxiliary guidance. Approaches include logit-level blending with small tuned experts (Liu et al., 2021; 2024; Khandelwal et al., 2020; Zhang et al., 2024b; Shen et al., 2024), distribution reshaping via learned value functions (Yang & Klein, 2021; Deng & Raffel, 2023; Mudgal et al., 2024; Han et al., 2024; Xu et al., 2025), and contrastive decoding or emulated fine-tuning (Li et al., 2023b; Mitchell et al., 2024). More recent methods apply this logit arithmetic dynamically, allowing a small specialized model to nudge the base model token-by-token using confidence-based gating (Fei et al., 2025; Li et al., 2024; Hu

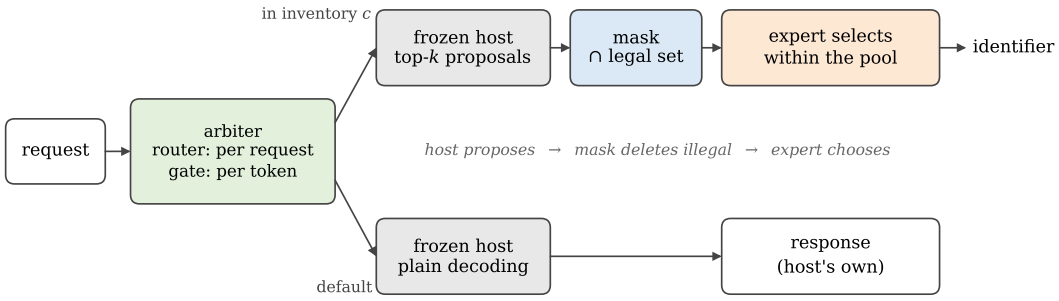

Figure 1: The pipeline: the router decides once per request; if an inventory is active, the mask and the expert act inside the host's sampler; the default class returns the bare host.

et al., 2026). All of them act token by token, and can fall below the host where the guiding model is ignorant. Recent works explore dynamic arbitration: CombLM and Co-LLM learn per-token combinations or deferral switches (Ormazabal et al., 2023; Shen et al., 2024), token-level routers select experts step-by-step (Xiong et al., 2026), and contrastive decoding with abstention makes per-request decisions using only the host (Kim et al., 2025). What remains open is arbitration for private-inventory guidance: formalizing exactly when a small expert improves or degrades a frozen host, and developing a routing rule that safely bypasses out-of-domain requests.

2.3 Request-level routing

A third paradigm investigates request-level routing, where cascades and learned routers select among complete models to optimize cost-quality trade-offs (Chen et al., 2024b; Ong et al., 2025; Ding et al., 2024; Lu et al., 2024; Chen et al., 2024c; 2026). The routing signal can derive from confidence tokens (Chuang et al., 2025), prefill activations (Varshney et al., 2026), or sub-query decomposers (Sridharan et al., 2025). Request-level selection also applies to decoding strategies (Su et al., 2026; Pan et al., 2026), though constraint-switching remains format-driven (Nguyen et al., 2026).

3 Adaptation in the Sampling Space

Our pipeline (Figure 1) modifies the sampling step to separate adaptation into three decisions, each with its own owner. Legality is handled by a **mask** (§3.2), knowledge by an **expert** (§3.3), and applicability, whether either applies to a request at all, by an arbiter (§3.4). The arbiter functions as either a token-level **gate** or a request-level **router**.

3.1 Problem setting and statement

The problem is to adapt a frozen host h to private inventories $C_1, \ldots, C_m$ without touching the host's internals. The host is served through its sampler: depending on the deployment, the user sees the next-token distribution of h (self-hosted) or a structured-outputs endpoint (closed API). Each inventory has a schema from which the set of legal identifiers can be derived.[1] Requests arrive as a mixed stream: an unknown fraction p_c concerns inventory c, and the remainder must be answered by the host as it is.

A deployed system serves the whole stream in one fixed configuration; its deployed accuracy is $A(p) = \sum_c p_c A_c + (1 - \sum_c p_c)\, A_{\text{def}}$, where A_c is its accuracy on requests concerning C_c and A_{def} its accuracy on the remainder. Because the deployment distribution p is unknown, the goal is Pareto improvement over the host: (A) improving A_c while (B) preserving A_{def} at the bare host's level, so that $A(p)$ is no worse than the host's accuracy for any mixture and improves whenever the mixture assigns positive mass to an inventory on which accuracy improves. We refer to this as the **two-condition contract**. Accordingly, we report every method as a triplet: accuracy on covered requests, accuracy on uncovered requests, and the bare host's baseline.

[1] "Private" means user-owned, never entering shared or provider-side training; the legal candidates and schema can travel in a request.

3.2 THE MASK: LEGALITY BY CONSTRUCTION

The mask is a deterministic filter on the host's next-token distribution. It activates at designated identifier slots, where an entry from the private inventory is required. At these slots, decoding is restricted to values permitted by the schema.[2] At each step, the mask removes illegal next tokens and renormalizes the remaining distribution. This provides a deterministic legality guarantee: when the mask is active, the emitted identifier is legal by construction. Its effect on accuracy is a separate question. Property 1 characterizes the value-level idealization (proof in Appendix A):

Property 1 (slot-level effect of the mask)**.** *Let $p_h(\cdot \mid x)$ denote the host's probability distribution over the possible values of the slot. Let $\hat{y} = \arg\max_v p_h(v|x)$ be its unconstrained prediction, and let $\hat{y}_L = \arg\max_{v \in L} p_h(v|x)$ be its most probable prediction restricted to the legal set L.*

The mask alters the emitted value if and only if the unconstrained choice is illegal ($\hat{y} \notin L$), in which case it outputs $\hat{y}_L$. Consequently, the mask's expected accuracy gain is $P(\hat{y} \notin L \wedge \hat{y}_L = y)$, where y is the gold value. This gain is nonnegative and upper-bounded by support recall at 1 ($\mathrm{SR@1} = P(\hat{y}_L = y)$). Since every masked output is forced into L, any resulting error is legal and invisible to legality checks.

In the value-level idealization, the mask improves accuracy when the host has *partial knowledge* of the inventory: its unconstrained prediction is illegal and its highest-ranked legal alternative is the gold value. When the host completely lacks domain knowledge, the mask yields negligible benefit. The governing metric is **support recall** ($\mathrm{SR@}k$). This is defined as the fraction of a probe set where the gold answer lies among the host's k most probable legal values. Because the practical mask operates token-by-token rather than over whole values, realized gains can depart from this bound in either direction. Gains can even become negative if metrics credit free-form strings instead of exact catalog keys. §4.2 quantifies this gap. For closed models, the mask relies on the provider's structured output APIs. In these systems, the legal set is passed as an enumerated list of allowed answers. This serves as a coarser, whole-answer variant of the same constraint. We evaluate both formulations in §4.2.

3.3 THE EXPERT: USER-OWNED KNOWLEDGE INSIDE THE HOST'S PROPOSAL SET

The domain knowledge absent from the host is supplied externally by the **expert**: a swappable network trained specifically to surface the correct inventory value. This expert can operate either as a standalone generator at the identifier slot, or as an auxiliary model that scores the host's candidates. With the mask, generation outside the slot stays with the host; without it, the expert votes among the host's candidates at every step.[3] For token-by-token coupling the expert shares the host's token family, so the two scores are commensurable at every step. At whole-candidate granularity this requirement lifts.[4] Our instantiation is a 0.5B language model with low-rank adapters (design rationale and training recipe in Appendix B). It can be one adapter per inventory (**expert** in §4) or one merged model serving several (**merged expert**). The merged expert's prefill representation can simultaneously serve as the router's input (§3.4).

For token-level coupling, both models score the next token. We merge these scores via a **coupling** rule following a strict principle: the host nominates, and the expert votes. The host proposes its k most probable tokens, and the expert votes within this pool, so that, outside the fallback below, an errant expert selects only plausible host candidates. The system optionally intersects this pool with the formal legal set, either before or after host nomination.[5] If no nominated candidate is legal, the pool widens to every legal continuation and the expert votes there; with the mask applied before nomination, the pool never empties.

Within the nominated pool of k candidates, final token scores are combined using a weighting coefficient β. We evaluate two regimes: $\beta = 1$, where both models contribute equally, and the limit $\beta \to \infty$, an "expert-decides" regime where the host acts solely as a candidate generator. Note that the

[2] We call this legal set the *rules*; it is derived from the schema and never from gold answers.

[3] A supervised retriever (Karpukhin et al., 2020; Reimers & Gurevych, 2019) trained on the same records lands 12 points below the bare expert (Appendix F.2).

[4] Experts from two other token families and a trained encoder match or beat the in-family expert (Appendix F.3).

[5] Both sequences produce empirically indistinguishable results. Appendix F defines and evaluates both approaches.

pool size k bounds the algorithmic behavior. At $k = 1$ with the mask applied first, the expert exerts no influence, recovering the masked host. Expanding k to the full vocabulary ($k = |V|$) removes the pool restriction: at $\beta = 1$ this is logit-sum (product-of-experts) decoding, DExperts without its anti-expert term, at $\beta \to \infty$ the expert alone.

We evaluate the full-vocabulary variant only as an ablation baseline. Restricting the expert to the host's k-candidate pool has three consequences:

- **Legality with the mask, plausibility, and a ceiling priced by k.** With the mask, the coupling inherits §3.2's legality guarantee; without it, the pool only keeps the expert among the host's plausible candidates, and illegal names remain possible (Table 4). The pool also prices a ceiling: a gold outside the host's k best legal candidates at a step cannot be emitted; at whole-candidate granularity (Appendix F.3) this caps accuracy at $\mathrm{SR@}k$.
- **Not safety.** When the expert confidently prefers a wrong but legal and plausible value (as on a sibling catalog it was never trained on), the pool does not stop it.
- **Not applicability.** β is fixed; whether the expert should act at all is the arbiter's decision (§3.4).

Section 4 uses short configuration labels defined in Appendix B. Our proposed method is "blend(k, β)", which combines host and expert scores over a candidate pool. We evaluate two variants: "expert-decides" (the expert makes the final choice) and "in-pool sum" (both models are weighted equally). All other labels are baselines, bounds, or ablations.

3.4 The applicability decision and deployment

The components so far supply legality and knowledge, but not the decision of when they apply. An always-on expert has no mechanism to satisfy condition (B) of §3.1: on a request outside every inventory, its vote is noise added to a competent host. Some arbiter must therefore switch the expert and the mask on and off, and the design space offers two granularities.

The gate decides per token. At each decoding step (or once per identifier slot, in the per-slot variant) a signal computed from the two models' current distributions decides whether the expert's vote counts at this position. We evaluate the natural signal families (raw confidence and the KL displacement of the tuned expert from its untuned twin), each per token or per slot, with and without the mask; definitions and threshold sweeps are in Appendix C.

The router decides once per request, before decoding begins. It functions as a multiclass classifier with per-class thresholds based on the expert's prefill state. It includes one class for each inventory and a default class that returns the request, as it arrived, to the unmodified host. We tune inventory thresholds for high precision. The highest-scoring class above its threshold wins; otherwise, the system defaults to the host. While token-level gating risks word-by-word failures, our request-level router is designed to make a single applicability decision before decoding begins. Section 4.3 evaluates whether step-by-step statistics can reliably make this applicability decision.

4 Experiments

4.1 Experimental setup

Evaluation Tasks. Except for the GSM8K control, the gold target belongs to a known set (a catalog, a controlled candidate set, or a schema), so legality and correctness are scored exactly.

- **API Catalogs (Gorilla):** TF Hub (688 questions, 80 legal identifiers), TorchHub (183, 94), and HuggingFace (893, 895). Metrics: Identifier match and name-level hallucination rate.
- **Function Calling (BFCL; 2,150 scored):** Candidate sets are controlled at size K, containing the gold target plus $K - 1$ distractors (random or BM25-hard (Robertson & Zaragoza, 2009)). Abstention is always legal and serves as the gold target for the irrelevance split. Metrics: Full-call correctness and name accuracy.
- **Semantic Parsing:** Text-to-SQL via BIRD (1,534 dev) and Text-to-Cypher via CypherBench (300 dev, 2,048 held-out). Here, the inventory acts as a schema and legality is defined by a formal grammar. Metric: Execution accuracy.
- **Out-of-Scope Control:** GSM8K (200 questions). Metric: Answer accuracy.

	TF Hub (688, K=80)	TorchHub (183, 94)	HF (893, 895)	BIRD EX (1,534)	CypherBench EX (2,048 held-out)	BFCL (K=10 hard)
host free-form	49.6	8.2	3.5	35.2	19.2	20.2 / 33.1
masked host, flat	**53.2**	6.6	2.2	**37.7**	n/a	**41.3** / 71.7
masked host, anchored	**55.7**	5.5	2.4	n/a	**21.3**	41.0
masked expert (host's anchor)	**58.9**	3.8	5.5	n/a	n/a	n/a
masked pair	**60.9** / **61.3** (flat / anch)	6.6	3.8	n/a	n/a	n/a
slot-level bound (Property 1)	6.1	0.0	0.4	n/a	n/a	n/a
invented names, free→masked	27.9→0	13.7→0	47.0→0	9.8→2.5	34.6→0.1	0.3→0

Table 1: Rules family performance on the Qwen2.5-7B host across six datasets. "Invented" indicates out-of-inventory hallucinations. "Bound" reflects the slot-level ceiling defined in Property 1. We detail all metrics in §4.1; EX: execution accuracy; BFCL cells: full-call / name accuracy; n/a: not defined on that dataset. Flat: the mask covers the whole legal set; anchored: the host emits the anchor (a domain or table name) first, and the mask covers only that anchor's names. The rows represent three distinct evaluation protocols (see Appendix B for baseline host scores).

host	TF free	TF masked (Δ, form)	HF free	HF masked (Δ)
Qwen2.5-7B	49.6	**55.7** (+6.1, anch)	3.5	2.4 (-1.1)
Qwen2.5-32B	33.3	**52.0** (+18.8, flat)	5.8	**3.5** (-2.4)
Qwen2.5-72B-AWQ	47.8	**51.7** (+3.9)	4.6	3.2 (-1.3)
Llama-3.2-1B	0.3	**16.7** (+16.4, anch)	0.3	0.9 (+0.6 n.s.)
Llama-3.3-70B-AWQ	35.2	**46.8** (+11.6, anch)	2.4	2.5 (+0.1 n.s.)
gpt-4o	46.4	**61.8** (+15.4, flat)	6.8	**14.7** (+7.8)
gpt-4.1	40.7	**64.1** (+23.4, flat)	7.4	**17.0** (+9.6)

Table 2: Identifier-match accuracy (%) on TF Hub (K=80) and HuggingFace (K=895), seven of the 21 hosts we test: free-form vs constrained decoding (best mask form; Δ in parentheses). The pattern holds for all 21; full grid, brackets, and the TorchHub column in Appendix D.

Statistical Reporting. We report performance differences using 95% paired bootstrap confidence intervals (2,000 resamples). Bold text indicates statistically significant results unless a caption states otherwise. Differences between accuracies are in percentage points. We also always list the legal-candidate set size (K) alongside masked scores.[6] Appendix B gives the full setup.

4.2 Where the rules pay

One rule across 21 hosts: the mask pays where the host partly knows the inventory. Table 1 reports the rules family under oracle routing; Table 2 tests the rule host by host. On TF Hub, where the hosts partially know the catalog, the mask improves 15 of the 16 open hosts' numbers (+3.9 to +22.2; four families, 1B–72B). Under structured outputs the mask lifts the frontier models to at best 64.1 (see Appendix D for the full results on the five API models). On HuggingFace, the catalog the open hosts do not know, no open host gains significantly (-2.4 to $+0.6$; the losses are mostly exact-key against identifier-metric gaps, Appendix D), while the frontier models, which know more of it, gain up to +9.6. In the anchored mask setting, where the host emits the anchor (a domain or table name) first, and then the mask covers only that anchor's names, efficacy still depends on the host's knowledge of the correct anchor: frontier models and the 32B host predict domains reliably and do better with the flat mask, while most open-weight hosts need the anchored narrowing. The pattern follows the host's knowledge, not the bed.

Defining the boundary across datasets. The remaining datasets isolate the limits of the mask's utility. On BIRD, where the schema is already included in the prompt, the mask adds minimal accuracy ($+2.5$ points $[+0.6, +4.2]$) but cuts invented names from 9.8% to 2.5% of queries. On CypherBench, the mask's benefit is purely structural: it improves execution accuracy by $+2.1$ points $[+1.4, +2.8]$ on the 2,048-question held-out set by forcing schema-legal names, but the per-slot candidate sets are too small for it to add meaningful factual knowledge. Finally, on BFCL, varying the candidate-set size (K) isolates the benefit of strict legality at large K. The sweep also shows a cost: forcing frontier models to choose from an enumerated list lowers their abstention accuracy by $\approx$16 points regardless of K (Appendix C); this fragility of abstention under forced choice is a known failure mode (Wen et al., 2025).

[6] Tables 4 and 5 run without the mask.

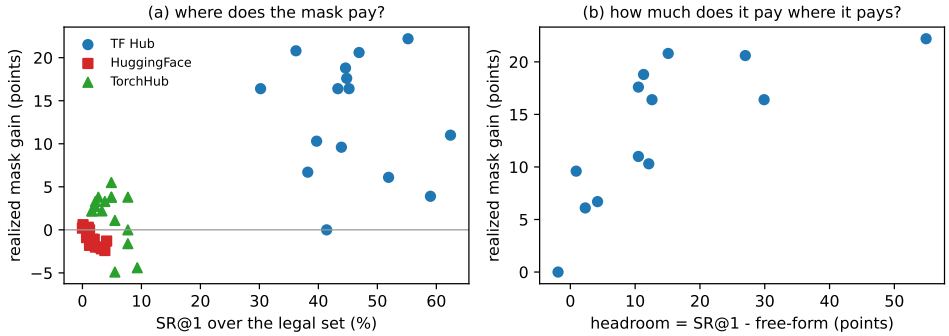

Figure 2: Support recall predicts where the mask pays and by how much. (a) Over 46 host–catalog pairs, the separation is largely between catalogs: every TF Hub pair has SR@1 $\geq 30\%$ and every HuggingFace and TorchHub pair SR@1 $\leq 10\%$. (b) Within TF Hub (13 bf16 hosts), raw SR@1 does not rank the gains; headroom does.

bed	zero-shot	+ anchored mask	merged expert + mask	router-composed (merged + mask)	+ dense cards (+500 tok)	cards + mask	cards + expert + mask
TF Hub	49.6	55.7	**67.4**	64.2	58.4	58.7	59.0*
HF	3.5	2.4	17.9	16.5	**22.7**	13.2	14.3*
TorchHub	8.2	5.5	13.1	11.5	18.0	**18.6**	12.0

Table 3: Prompt-space versus sampling-space adaptation (Qwen2.5-7B, identifier-match accuracy %). Best performance is bolded. "Cards" inserts the top-5 dense-retrieved APIs into the prompt. "Cards + Mask" and "Cards + Expert + Mask" apply our methods to those prompted baselines (*: $n = 300$, see Appendix F for details). All extended baselines and sample sizes are provided in the appendix.

cell	k=5	k=10	k=25	k=50	k=200
expert-decides (halluc %)	65.0 (5.7)	67.0 (2.3)	67.7 (0.7)	69.3 (0.0)	69.8 (0.3)
in-pool sum	63.7	65.3	65.7	64.0	63.8
KL-gated expert-decides, τ=0.3 (halluc %)	57.3 (16.0)	60.0 (14.3)	60.0 (10.3)	64.3 (6.7)	67.0 (6.3)
KL-gated expert-decides, τ=0.8	52.0	54.7	55.3	58.3	59.3

Table 4: The coupling substrate on the expert's catalog (TF Hub; 7B host + 0.5B log expert; accuracy %, hallucination % in parentheses). References: host 49.1, bare expert 68.8, proxy 64.0, on all 688 questions; the expert-decides and τ=0.3 cells at $k \leq 50$ use the first 300 of them (host 48.0 to 48.3).

Predicting mask utility via support recall. Across 16 open-weight hosts (1B–72B) and three catalogs (Appendix D), support recall (SR) robustly predicts mask efficacy. Across 46 evaluated pairs, the mask never degrades performance when SR@1 $\geq 30\%$, while all regressions occur when SR@1 $\leq 10\%$ (Pearson $r = 0.77$; Figure 2a). Below 10%, the observed changes stay within ± 5.5 points. On TF Hub, where the mask is beneficial, gain magnitudes correlate strongly (Spearman $\rho = 0.82$; 13 `bf16` hosts; Figure 2b) with model *headroom* (SR@1 minus free-form accuracy). For example, Llama-3.2-1B achieves only 0.3% free-form accuracy but ranks the correct legal candidate first in 30% of cases; the mask recovers part of this latent knowledge, improving accuracy by 16.4 percentage points. On TorchHub, twelve of sixteen models improve despite SR@1 $\leq 10\%$ (though minor gains are insignificant after Holm correction; Appendix D).

Comparison against retrieval. We benchmark our sampling-space strategies against standard prompt retrieval (Table 3).[7] On TF Hub, the masked merged expert outperforms retrieval by 9.0 points. However, prompt retrieval remains superior on HuggingFace and TorchHub. On HuggingFace, applying the mask over retrieved context degrades performance by 9.5 points (versus a $+0.5$ gain on TorchHub). Adding the expert to prompts containing retrieved cards does not improve accuracy and reduces it on TorchHub. The expert was trained without retrieved-card context (Appendix F). This divergence likely stems from the limits of our training regime's capacity: prompt retrieval leverages explicit non-parametric matching, while the expert is limited to parametric recall.

[7]Expert and router-composed settings pair against their host baselines (49.0 / 3.6 / 7.7, Appendix B), yielding $+15.3$ $[+12.4, +18.2]$ on TF Hub and $+3.8$ $[-0.5, +8.2]$ on TorchHub.

method (setting)	TF own	HF questions	Torch questions	GSM8K
expert-decides always-on (k=200 on TF Hub, 50 elsewhere)	+20.6	−3.3	−7.7	−32.0
logit sum (in-pool on TF Hub; full-vocabulary elsewhere)	+14.7	−1.3	−3.3	−10.5
proxy tuning	+14.9†	−2.0	−2.2	−1.5
proxy-in-pool (κ=0, k=25)	+15.0	−2.0	−2.2	+0.5
raw-conf gate (τ=0.8)	+16.4	−1.3	−1.6	−6.0
KL-gated expert-decides (0.8 ∩ pool50)	+9.7	−2.0	−6.0	+0.0
learned per-token gate (head on the expert's state)	+14.7	−1.7	0.0 (floor)	+0.0
router → coupling, deployed (multiclass, defaulted to host) expert-decides in the host's top-200, n=688	**+15.7**	**+0.0** (= host)	**+0.0** (= host)	**+0.0** (exact)
router → bare expert (0.5B alone, no host pool; same verdicts, n=688)	+14.2	+0.0	+0.0	+0.0

Table 5: Results of the two-condition test: accuracy changes against the bare host, TF Hub expert in every row. Condition A is the expert's own dataset, Condition B three datasets outside its inventory. The bold row is our deployed method (the coupling, no mask; the abstract's 16 points). The learned gate is fitted and scored on disjoint halves of the questions ($n = 150$ on its catalog), so its gap to the router is suggestive. A default verdict returns the host's answer to the request under its own prompt (Appendix D). †: no retained per-question records; floor: the host itself scores 0.0 there (Appendix D). Full confidence intervals in Appendix D.

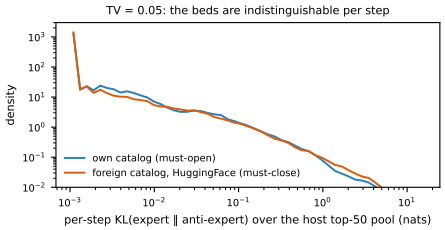

Figure 3: Per-step KL displacement of the expert from its untuned twin. We compared these changes on a familiar target dataset versus an unfamiliar foreign dataset. The two distributions overlap almost entirely: a large change does not signal that the expert should act.

4.3 The applicability decision: gates against the router

This section investigates whether expert activation can be controlled on a per-token basis. No scalar per-step signal meets both conditions of our two-condition test at one setting (Table 5). A learned per-token classifier trained on the expert's hidden states recovers most of the performance gain. However, it still trails the request-level router and incorrectly activates on foreign catalogs.

T1: Scalar per-token gates fail; a learned per-token head trails the router. We evaluated KL divergence gates per-token and per-slot (thresholds 0.3–4.0 nats; masked and unmasked; pool sizes 5–200). On the expert's native catalog, thresholds permissive enough to trigger the expert still degrade performance by 4.5 points compared to an always-on, ungated baseline. On out-of-domain catalogs (e.g., HuggingFace), these same thresholds falsely trigger, emitting out-of-domain identifiers in 78–79% of cases. We could not find any viable threshold; below 1.5 nats the gate triggers indiscriminately; above it, a gate that decides once per identifier slot never opens, and one that decides at every token keeps at most a third of the gain. Raw confidence, the signal CoGate gates on (Hu et al., 2026), behaves the same way (optimal τ varies from 0.5 to 0.95 by dataset).

Proposition 1 (limit of per-step statistic gates)**.** *Let $S(x)$ be any per-step statistic and let g be* any *decision rule that reads only S; thresholds are the special case g_τ = "the expert decides iff $S \geq \tau$".*

For every such g, (share of must-open steps g closes) + (share of must-close steps it opens) $\geq 1 - \mathrm{TV}(P_S^A, P_S^B)$*, where P_S^A and P_S^B are the distributions of S on the two kinds of steps and* TV *is the total variation distance. This is Le Cam's two-point bound for gates: it covers rules that read one scalar at one step, not rules that read position, history, or the full state; Table 5 tests one of these, a learned gate on the expert's state. (Proof in Appendix A.)*

The failure of threshold gating stems directly from the underlying signal distributions. Across 19,200 decoding steps per testbed, in-domain and out-of-domain signals overlap almost entirely (medians 0.041 vs. 0.045 nats; Figure 3). On the recorded decodes the binned TV is 0.05, so by Proposition 1 the two error shares of any rule reading the binned signal sum to at least 0.95, an equal-prior error

of at least 47.5%, barely better than chance. Averaging the signal across the request or restricting it to identifier tokens lowers that floor only to 0.38–0.91 (Appendix C).

T2: The learned routing signal succeeds and scales. A logistic head on the merged expert's prefill state isolates the routing signal without emitting tokens. For binary routing, it accurately separates domains (F1 82.7, AUC 0.93), retaining 67.3 of the expert's 69.8 always-on score while defaulting to the host on HuggingFace and arithmetic testbeds. This extends to a multiclass router (multiple inventories plus a default fallback), which trails the always-on coupling by 3.0 points on the 200-question suite (4.9 on the full dataset, Appendix E). The default class is sensitive to the types of general requests used in training. Without the inventory-specific serving frame, neither expert-prefill head reliably recognizes TF Hub or HuggingFace requests (Appendix F). A segment-level variant recovers 72–91% of oracle gains on mixed multi-expert queries (Appendix G), and results replicate on the Falcon3 family. Ablations show an untuned-twin control loses 14 to 31 points of catalog activation and 5 composed points. However, a nearest-neighbor router over generic sentence embeddings nearly matches the tuned head on TF Hub and general traffic, indicating that tuning primarily improves HuggingFace recall rather than establishing the baseline routing signal (Appendix F). Finally, on the sibling TorchHub dataset the router recognizes 77% of requests and sends none to TF Hub (Appendix F).

T3: Portability to closed-API models. The evaluated closed APIs accept a logit bias and structured outputs but not a neural expert, so we execute the pipeline serially. A local router first determines the domain, then passes its verdict and the legal candidate set to the closed host via standard structured output constraints. On a mixed stream of 1,764 requests, it keeps all five tested frontier models within 0.6 points of a perfect oracle router (Appendix F). For four of the five models, it gains +3.8 to +8.0 points over free-form generation; gpt-4.1-mini is unchanged.

4.4 Gain–loss accounting and regret

We use measured activation rates and average strategy gains and losses to approximate the effect of routing on accuracy. Appendix E states the required independence assumption.

- **Gain–loss accounting:** Per-class thresholds and a default fallback account for the routing loss and the captured gain on the datasets where activation was measured. The in-inventory shortfall (4.9 on the full dataset, 3.0 on the suite) is within 1.5 points of the capture relation's 4.7 and 4.5.
- **Minimized regret:** Against an oracle on the 200-question suite, the best scalar gate at one setting on the common suite leaves 13.2 points of regret. Our router leaves 3.0 points.

The accounting relations and the full regret table are in Appendix E.

Limitations. The router depends on its training and serving distribution: its default class protects only the kinds of general requests it was fitted on, so under a hub frame it misroutes 14–99.5% of unseen general requests (17–20% of an unseen chat dataset after a refit with broad negatives), and without the serving frame, it does not recognize its own catalog (Appendix F). Without usage logs the expert adds little over the mask and cannot route itself (Appendix B). On closed APIs only the verdict and the legal set transfer; the expert's guidance does not.

5 Conclusion

In this paper, we explore how to adapt a frozen, served language model to a private inventory entirely in the sampling space, decomposing the problem into three distinct responsibilities: legality, knowledge, and applicability. First, we find that legality belongs to the sampler: constrained decoding extracts latent knowledge that support recall predicts, allowing utility to be predicted offline. Second, a user-owned expert can supply the missing knowledge, but collapses outside its domain. Finally, we find that request-level representations provide an effective applicability signal in the evaluated settings: because thresholded per-step signals do not separate domains (Proposition 1 bounds every scalar gate by the measured overlap) and a learned per-token head recovers most of the gain but still trails the request-level router, we propose a request-level routing mechanism over the prefill state. This router keeps its losses within its measured activation rates on the datasets its default class was fitted to, tracks an oracle closely on the evaluated streams, and transfers its verdict and mask to closed-API environments.

Reproducibility statement

All evaluated datasets are public benchmarks (Gorilla, BFCL, BIRD, CypherBench, GSM8K), with the exception of the generated beds reported in the appendix (the ShadowHub private catalog, the paraphrased evaluation bed, and the mixed-request beds), which we release together with the scripts that produced them. We build every mask from public schemas or catalogs, never from the gold answers. Decoding is greedy and deterministic (one sampling setting is reported in Appendix D). Reference cells reproduce exactly across runs, and within a ± 0.5 margin across different hardware. We compute every reported number directly from per-question decode records. To guarantee reproducibility, we release these records alongside our evaluation code, expert training manifests, and exact prompts (protocol in Appendix B). We also provide the timestamped documents used to pre-register the two-condition test and pool-size predictions. Paired bootstrap confidence intervals are computed over these exact records. Finally, all experts are trained with fixed seeds, and hosts run frozen in bf16 (4-bit AWQ where stated) on A100, H100, and A40 GPUs.

Statement on the use of large language models

We used large language models under author supervision, for the purposes declared on the submission form.

- **Research execution.** The research questions, the experimental design, and every claim are the authors'. We used assisted coding to implement the evaluation and training code, and the analysis scripts; the authors audited each reported number against the per-question decode records described above.
- **Synthetic data.** We strictly exclude model-generated data from all main text results. We limit synthetic data to three specific appendix evaluations (Appendix B.1): a cold-start regime with host-generated training questions, the ShadowHub catalog with synthetic logs, and a paraphrased evaluation set. We rigorously measure and report the data overlap between all generated training records and evaluation questions in Appendix B.1.
- **Mathematical claims.** We used the LLM to edit the mathematical proof for Proposition 1, and to identify Le Cam's two-point bound.
- **Literature discovery.** We used AI-assisted search to find possible gaps in the literature review we had; the authors read and verified every cited paper.
- **Writing.** An assistant edited and polished the manuscript.

The authors take full responsibility for the content of this paper, including every claim, number, and citation.

References

Lakshya A. Agrawal, Aditya Kanade, Navin Goyal, Shuvendu K. Lahiri, and Sriram Rajamani. Monitor-guided decoding of code LMs with static analysis of repository context. In *NeurIPS*, 2023.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self reflection. In *ICLR*, 2024.

Debangshu Banerjee, Tarun Suresh, Shubham Ugare, Sasa Misailovic, and Gagandeep Singh. CRANE: Reasoning with constrained LLM generation. In *ICML*, 2025.

Luca Beurer-Kellner, Marc Fischer, and Martin Vechev. Guiding llms the right way: Fast, non-invasive constrained generation. In *ICML*, 2024.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Shuaichen Chang, Jun Wang, Mingwen Dong, Lin Pan, Henghui Zhu, Alexander Hanbo Li, Wuwei Lan, Sheng Zhang, Jiarong Jiang, Joseph Lilien, Steve Ash, William Yang Wang, Zhiguo Wang, Vittorio Castelli, Patrick Ng, and Bing Xiang. Dr.Spider: A diagnostic evaluation benchmark towards text-to-SQL robustness. In *ICLR*, 2023.

Lingjiao Chen, Matei Zaharia, and James Zou. How is ChatGPT's behavior changing over time? *Harvard Data Science Review*, 2024a. arXiv:2307.09009.

Lingjiao Chen, Matei Zaharia, and James Zou. FrugalGPT: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024b. arXiv:2305.05176.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-based router by dual contrastive learning for assembling large language models. In *NeurIPS*, 2024c.

Zhijun Chen, Xiaodong Lu, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Ming Li, Likang Xiao, Dingqi Yang, Xiao Huang, Yikun Ban, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on LLM ensemble. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2026. Survey Track; arXiv:2502.18036.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to route LLMs with confidence tokens. In *ICML*, 2025. arXiv:2410.13284.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. In *ICLR*, 2021.

Haikang Deng and Colin Raffel. Reward-augmented decoding: Efficient controlled text generation with a unidirectional reward model. In *EMNLP*, 2023.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor R"uhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-efficient and quality-aware query routing. In *ICLR*, 2024.

Yixin Dong, Charlie F. Ruan, Yaxing Cai, Ruihang Lai, Ziyi Xu, Yilong Zhao, and Tianqi Chen. XGrammar: Flexible and efficient structured generation engine for large language models. In *MLSys*, 2025. arXiv:2411.15100.

Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson. From local to global: A graph RAG approach to query-focused summarization. *arXiv preprint arXiv:2404.16130*, 2024.

Rajaa El Hamdani, Samy Haffoudhi, Nils Holzenberger, Fabian Suchanek, Thomas Bonald, and Fragkiskos D. Malliaros. Retrieval-constrained decoding reveals underestimated parametric knowledge in language models. *arXiv preprint arXiv:2509.23417*, 2025.

European Data Protection Board. Opinion 28/2024 on certain data protection aspects related to the processing of personal data in the context of AI models. Technical report, EDPB, 2024.

Yu Fei, Yasaman Razeghi, and Sameer Singh. Nudging: Inference-time alignment of LLMs via guided decoding. In *ACL*, 2025. arXiv:2410.09300.

Yanlin Feng, Simone Papicchio, and Sajjadur Rahman. CypherBench: Towards precise retrieval over full-scale modern knowledge graphs in the LLM era. In *ACL*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Qian-Yu Guo, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2023.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *NeurIPS*, 2017.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. Grammar-constrained decoding for structured NLP tasks without finetuning. In *EMNLP*, 2023.

Yu Gu and Yu Su. ArcaneQA: Dynamic program induction and contextualized encoding for knowledge base question answering. In *COLING*, 2022.

Yu Gu, Sue Kase, Michelle Vanni, Brian Sadler, Percy Liang, Xifeng Yan, and Yu Su. Beyond I.I.D.: Three levels of generalization for question answering on knowledge bases. In *WWW*, 2021.

Seungwook Han, Idan Shenfeld, Akash Srivastava, Yoon Kim, and Pulkit Agrawal. Value augmented sampling for language model alignment and personalization. *arXiv preprint arXiv:2405.06639*, 2024.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022.

Minghao Hu, Lannan Luo, Allen Roush, and Phillip Howard. CoGate: Confidence-gated co-decoding for secure code generation. *arXiv preprint arXiv:2607.28529*, 2026.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *EMNLP*, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models. In *ICLR*, 2020.

Hyuhng Joon Kim, Youna Kim, Sang-goo Lee, and Taeuk Kim. When to speak, when to abstain: Contrastive decoding with abstention. In *ACL*, 2025. arXiv:2412.12527.

Lucien Le Cam. Convergence of estimates under dimensionality restrictions. *The Annals of Statistics*, 1(1):38–53, 1973.

Ivan Yee Lee, Loris D'Antoni, and Taylor Berg-Kirkpatrick. The format tax. In *COLM*, 2026. arXiv:2604.03616.

Dong Li, Meng Yan, Yaosheng Zhang, Zhongxin Liu, Chao Liu, Xiaohong Zhang, Ting Chen, and David Lo. CoSec: On-the-fly security hardening of code LLMs via supervised co-decoding. In *ISSTA*, 2024. doi: 10.1145/3650212.3680371.

Fangzheng Li, Aimin Zhang, and Chen Lv. Constraint tax in open-weight LLMs: An empirical study of tool calling suppression under structured output constraints. *arXiv preprint arXiv:2606.25605*, 2026.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.-C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? a BIg bench for large-scale database grounded text-to-SQLs. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks*, 2023a.

Xiang Lisa Li, Ari Holtzman, Daniel Fried, Percy Liang, Jason Eisner, Tatsunori Hashimoto, Luke Zettlemoyer, and Mike Lewis. Contrastive decoding: Open-ended text generation as optimization. In *ACL*, 2023b.

Chen Ling, Xujiang Zhao, Jiaying Lu, Chengyuan Deng, Can Zheng, Junxiang Wang, Tanmoy Chowdhury, Yun Li, Hejie Cui, Xuchao Zhang, Tianjiao Zhao, Amit Panalkar, Dhagash Mehta, Stefano Pasquali, Wei Cheng, Haoyu Wang, Yanchi Liu, Zhengzhang Chen, Haifeng Chen, Chris White, Quanquan Gu, Jian Pei, Carl Yang, and Liang Zhao. Domain specialization as the key to make large language models disruptive: A comprehensive survey. *ACM Computing Surveys*, 58(3), 2025. arXiv:2305.18703.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *ACL*, 2021.

Alisa Liu, Xiaochuang Han, Yizhong Wang, Yulia Tsvetkov, Yejin Choi, and Noah A. Smith. Tuning language models by proxy. In *COLM*, 2024.

Keming Lu, Hongyi Yuan, Runji Lin, Junyang Lin, Zheng Yuan, Chang Zhou, and Jingren Zhou. Routing to the expert: Efficient reward-guided ensemble of large language models. In *NAACL*, 2024.

Daniel Maninger, Leon Chemnitz, Jannis Brugger, Tushar Lamba, Amir Molzam Sharifloo, and Mira Mezini. Mitigating errors in LLM-generated web API invocations via retrieval-augmented generation and constrained decoding. *ACM Transactions on Software Engineering and Methodology*, 37 (4), 2026. arXiv:2607.05936.

Menlo Ventures. The state of generative AI in the enterprise. Technical report, Menlo Ventures, 2024.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D. Manning. An emulator for fine-tuning large language models using small language models. In *ICLR*, 2024. arXiv:2310.12962.

Sidharth Mudgal, Jong Lee, Harish Ganapathy, YaGuang Li, Tao Wang, Yanping Huang, Zhifeng Chen, Heng-Tze Cheng, Michael Collins, Jilin Chen, Alex Beutel, and Ahmad Beirami. Controlled decoding from language models. In *ICML*, 2024.

Ngoc Trinh Hung Nguyen, Alonso Silva, Laith Zumot, Liubov Tupikina, Armen Aghasaryan, and Mehwish Alam. Thinking before constraining: A unified decoding framework for large language models. In *Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing (EMNLP): Industry Track*, 2026. To appear; arXiv:2601.07525.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M. Waleed Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs with preference data. In *ICLR*, 2025.

OpenAI. GPT-4 technical report. Technical report, OpenAI, 2023. arXiv:2303.08774.

OpenAI. Introducing structured outputs in the API. `https://openai.com/index/introducing-structured-outputs-in-the-api/`, 2024.

Aitor Ormazabal, Mikel Artetxe, and Eneko Agirre. CombLM: Adapting black-box language models through small fine-tuned models. In *EMNLP*, 2023.

Zhihong Pan, Kai Zhang, Yuze Zhao, and Yupeng Han. Route to reason: Adaptive routing for LLM and reasoning strategy selection. In *Proceedings of the ACM Web Conference (WWW)*, 2026. arXiv:2505.19435.

Kanghee Park, Jiayu Wang, Taylor Berg-Kirkpatrick, Nadia Polikarpova, and Loris D'Antoni. Grammar-aligned decoding. In *NeurIPS*, 2024.

Kanghee Park, Timothy Zhou, and Loris D'Antoni. Flexible and efficient grammar-constrained decoding. In *ICML*, 2025. arXiv:2502.05111.

Shishir G. Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. Gorilla: Large language model connected with massive APIs. In *NeurIPS*, 2024. arXiv:2305.15334.

Shishir G. Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. Berkeley function calling leaderboard (BFCL). `https://gorilla.cs.berkeley.edu/leaderboard.html`, 2025. v3 splits: simple, multiple, live, irrelevance.

Yujia Qin, Shengding Hu, Yankai Lin, Weize Chen, Ning Ding, Ganqu Cui, Zheni Zeng, Yufei Huang, Chaojun Xiao, Chi Han, et al. Tool learning with foundation models. *ACM Computing Surveys*, 57 (4), 2024. arXiv:2304.08354.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *EMNLP-IJCNLP*, 2019.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

AmirHossein Safdarian, Milad Mohammadi, Ehsan Jahanbakhsh Bashirloo, Mona Shahamat Naderi, and Heshaam Faili. SchemaGraphSQL: Efficient schema linking with pathfinding graph algorithms for text-to-SQL on large-scale databases. In *Findings of the Association for Computational Linguistics: EACL*, 2026. arXiv:2505.18363.

Henry Scheffé. A useful convergence theorem for probability distributions. *The Annals of Mathematical Statistics*, 18(3):434–438, 1947.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *EMNLP*, 2021.

Shannon Zejiang Shen, Hunter Lang, Bailin Wang, Yoon Kim, and David Sontag. Learning to decode collaboratively with multiple language models. In *ACL*, 2024.

Shrihari Sridharan, Sourjya Roy, Anand Raghunathan, and Kaushik Roy. Experts are all you need: A composable framework for large language model inference. *arXiv preprint arXiv:2511.22955*, 2025.

Chloe H. Su, Zhe Ye, Samuel Tenka, Aidan Yang, Soonho Kong, and Udaya Ghai. Learning adaptive LLM decoding. *arXiv preprint arXiv:2603.09065*, 2026.

Zhi Rui Tam, Cheng-Kuang Wu, Yi-Lin Tsai, Chieh-Yen Lin, Hung-yi Lee, and Yun-Nung Chen. Let me speak freely? a study on the impact of format restrictions on performance of large language models. In *EMNLP (Industry)*, 2024.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. Graph neural prompting with large language models. In *AAAI*, 2024.

Alexandre B. Tsybakov. *Introduction to Nonparametric Estimation*. Springer Series in Statistics. Springer, New York, 2009.

Alexander Matt Turner, Lisa Thiergart, David Udell, Gavin Leech, Ulisse Mini, and Monte MacDiarmid. Steering language models with activation engineering. In *ICLR*, 2025. arXiv:2308.10248.

Shubham Ugare, Tarun Suresh, Hangoo Kang, Sasa Misailovic, and Gagandeep Singh. SynCode: Llm generation with grammar augmentation. *Transactions on Machine Learning Research*, 2025. arXiv:2403.01632.

Tanay Varshney, Annie Surla, Michelle Xu, Gomathy Venkata Krishnan, Maximilian Jeblick, David Austin, Neal Vaidya, and Davide Onofrio. LLM router: Rethinking routing with prefill activations. In *COLM*, 2026. arXiv:2603.20895.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *ACL*, 2020.

Renxi Wang, Xudong Han, Lei Ji, Shu Wang, Timothy Baldwin, and Haonan Li. ToolGen: Unified tool retrieval and calling via generation. In *ICLR*, 2025.

Bingbing Wen, Jihan Yao, Shangbin Feng, Chenjun Xu, Yulia Tsvetkov, Bill Howe, and Lucy Lu Wang. Know your limits: A survey of abstention in large language models. *Transactions of the Association for Computational Linguistics*, 13:529–556, 2025.

Rui Wen, Lu Sun, Jiayang Liu, Zesheng Xu, Tianshuo Cong, and Zheng Li. The benchmark illusion: Pruned LLMs can pass multiple choice but fail to answer. *arXiv preprint arXiv:2606.17609*, 2026.

Brandon T Willard and Rémi Louf. Efficient guided generation for large language models. *arXiv preprint arXiv:2307.09702*, 2023.

Nuoya Xiong, Yuhang Zhou, Hanqing Zeng, Zhaorun Chen, Furong Huang, Shuchao Bi, Lizhu Zhang, and Zhuokai Zhao. Token-level LLM collaboration via FusionRoute. In *ICML*, 2026. arXiv:2601.05106.

Yuancheng Xu, Udari Madhushani Sehwag, Alec Koppel, Sicheng Zhu, Bang An, Furong Huang, and Sumitra Ganesh. GenARM: Reward guided generation with autoregressive reward model for test-time alignment. In *ICLR*, 2025.

Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In *NAACL*, 2021.

Yifan Yao, Jinhao Duan, Kaidi Xu, Yuanfang Cai, Zhibo Sun, and Yue Zhang. A survey on large language model security and privacy. *High-Confidence Computing*, 2024.

Hanning Zhang, Shizhe Diao, Yong Lin, Yi R. Fung, Qing Guan, Yujiang Liang, Tong Li, Yue Zhang, William Yang Wang, and Tong Zhang. R-Tuning: Instructing large language models to say 'i don't know'. In *NAACL*, 2024a.

Kaiyan Zhang, Jianyu Wang, Ermo Hua, Biqing Qi, Ning Ding, and Bowen Zhou. CoGenesis: A framework collaborating large and small language models for secure context-aware instruction following. In *ACL*, 2024b.

Kexun Zhang, Hongqiao Chen, Lei Li, and William Yang Wang. Syntax error-free and generalizable tool use for LLMs via finite-state decoding. In *ICML*, 2024c. arXiv:2310.07075.

A PROOFS

A.1 PROOF OF PROPERTY 1

Proof. Work in the value-level idealization, in which values are atomic and decoding is greedy under a fixed tie-breaking rule. Let

- V be the finite universe of complete values, and $L \subseteq V$ the legal set fixed by the schema;
- $p_h(\cdot \mid x)$ be the host's distribution over V for the request x;
- $\hat{y} := \arg\max_{v \in V} p_h(v|x)$ be the unconstrained prediction, and $\hat{y}_L := \arg\max_{v \in L} p_h(v|x)$ the prediction under the mask;
- $y \in L$ be the gold value.

We want to prove three things: *(i)* the mask changes the emitted value exactly when $\hat{y} \notin L$; *(ii)* its accuracy gain is $\Delta = P(\hat{y} \notin L \wedge \hat{y}_L = y)$, which lies in $[0, \mathrm{SR@1}]$; and *(iii)* every error the mask leaves behind is a legal value.

(i) When the mask acts. If $\hat{y} \in L$, then $\hat{y}$ maximizes $p_h(\cdot|x)$ over $V \supseteq L$ and therefore over L as well, so the same tie-break gives $\hat{y}_L = \hat{y}$. If $\hat{y} \notin L$, then $\hat{y}_L \neq \hat{y}$: by construction $\hat{y}_L$ is an element of L while $\hat{y}$ is not, and a value inside L cannot equal one outside it. Hence $\{\hat{y}_L \neq \hat{y}\} = \{\hat{y} \notin L\}$: the mask alters the emitted value exactly when the unconstrained choice is illegal, and then emits $\hat{y}_L$.

(ii) Gain. Since $y \in L$, $\hat{y} = y$ forces $\hat{y} \in L$ and hence $\hat{y}_L = y$ by (i): the mask never turns a correct answer wrong. So $\{\hat{y} = y\} \subseteq \{\hat{y}_L = y\}$, the two events differ only where $\hat{y} \notin L$, and

$$\Delta \;:=\; P(\hat{y}_L = y) - P(\hat{y} = y) \;=\; P(\hat{y} \notin L \wedge \hat{y}_L = y) \;\geq\; 0,$$

which is at most $P(\hat{y}_L = y) = \mathrm{SR@1}$.

(iii) Silent errors. The masked output is $\hat{y}_L \in L$ by construction, so an incorrect masked output is a legal value; no legality check can flag it. ■

Remark. Token-level masking relaxes exactly the idealization declared above, which is how a realized gain can depart from the bound in either direction, and fall below zero where the metric credits a string that is not a catalog key; Appendix D traces most of the measured losses to free-form outputs the identifier metric credits but that are not catalog keys, which a prefix trie cannot reproduce. Figure 4 gives a worked example of both regimes of Property 1.

(a) the host has partial knowledge: a near-miss, converted
Q: "Show me the right API call to generate image feature vectors using MobileNet V2 ..."

host, free-form: `tfhub.dev/tensorflow/mobilenet_v2/feature_vector`
invented: no such catalog entry

mask + expert: `tfhub.dev/google/imagenet/mobilenet_v2_100_224/feature_vector`
the catalog entry = gold

the invented name is a corruption of the gold; legality + the expert recover it

(b) the host is clueless: legal-but-wrong, silent
Q: "The user is interested in a tool to find relationships between medical terms."

host, free-form: `microsoft/deberta-v3-large-mnli`
invented: plausible-looking, not in the catalog

mask + expert: `dslim/bert-base-NER-uncased`
legal, but wrong (gold: GanjinZero/UMLSBert_ENG)

every legality check passes; the error is silent

Figure 4: Worked example, real instances from the decode records: (a) the host has partial knowledge: its invented URL is a corruption of the gold, which mask + expert recover; (b) the host is clueless: the mask forces a legal-but-wrong answer that no legality check can flag.

A.2 PROOF OF PROPOSITION 1

Intuition. At each decoding step, the system makes a binary decision: activate the expert or rely on the host. We define a step as *must-open* if the expert should decide it (a request on its catalog), and *must-close* if the host should (a request outside it).

A *gate* acts as a lightweight proxy for this decision. It computes a per-step scalar statistic and opens if this value crosses a threshold. Any such gate suffers from two types of errors:

- α: the fraction of must-open steps it incorrectly closes (sacrificing the expert's gain).
- β: the fraction of must-close steps it incorrectly opens (corrupting a correct host answer).

(Note that this error rate β is distinct from the blend weight in Section 3.)

Proposition 1 establishes a tight lower bound on these combined errors:

$$\alpha + \beta \;\geq\; 1 - \mathrm{TV}$$

Total Variation (TV) measures how differently the statistic behaves across the two types of steps. In short: a gate is only as effective as the signal it reads. If the statistic looks identical whether the expert is needed or not ($\mathrm{TV} = 0$), the error floor is 1: equivalent to blind guessing. If the signal separates the cases perfectly ($\mathrm{TV} = 1$), the floor is 0, meaning a perfect gate exists.

Proof. Let

- S be the per-step statistic and P_S^A, P_S^B its laws on must-open and must-close steps;
- a *rule* be a measurable $g : \mathbb{R} \to \{\text{open}, \text{close}\}$, identified with its open set $E_g := \{s : g(s) = \text{open}\}$, so that $\alpha(g) = 1 - P_S^A(E_g)$ and $\beta(g) = P_S^B(E_g)$, and the *threshold* rule g_τ be the case $E_{g_\tau} = [\tau, \infty)$;
- F_A, F_B be the CDFs of P_S^A, P_S^B, and $\mathrm{TV} := \sup_E |P_S^A(E) - P_S^B(E)|$ over measurable $E \subseteq \mathbb{R}$, $\mathrm{KS} := \sup_t |F_A(t) - F_B(t)|$;
- $\mu := P_S^A + P_S^B$ with densities $p_A := dP_S^A/d\mu$, $p_B := dP_S^B/d\mu$, and $E^* := \{s : p_A(s) > p_B(s)\}$.

We want to prove three things: *(i)* every rule that reads only S satisfies $\alpha(g) + \beta(g) \geq 1 - \mathrm{TV}$, and that floor is exactly attained; *(ii)* over half-line rules of either orientation the tight constant is $1 - \mathrm{KS}$; and *(iii)* allowing randomization does not lower it.

(i) Claim: every rule that reads only S satisfies $\alpha(g) + \beta(g) \geq 1 - \mathrm{TV}$, and some rule attains it. For every g, $\alpha(g) + \beta(g) = 1 - \big(P_S^A(E_g) - P_S^B(E_g)\big)$, and E_g is one admissible event in the supremum defining TV, so $\alpha(g) + \beta(g) \geq 1 - \mathrm{TV}$, which is the inequality. It is attained: Scheffé's identity (Scheffé, 1947) gives $\mathrm{TV} = P_S^A(E^*) - P_S^B(E^*)$, which the likelihood-ratio rule $g^* := \mathbb{1}[E^*]$ attains. Hence

$$\inf_g \big(\alpha(g) + \beta(g)\big) \;=\; 1 - \mathrm{TV}(P_S^A, P_S^B),$$

Le Cam's two-point identity (Le Cam, 1973; Tsybakov, 2009), whose right-hand side is twice the equal-prior Bayes error of A vs. B given S. Both halves of the claim are established.

(ii) Claim: over threshold rules the tight constant is $1 - \mathrm{KS}$, and $\mathrm{KS} \leq \mathrm{TV}$, so thresholds are the weaker family. Write $F(\tau^-) := \lim_{t \uparrow \tau} F(t) = P(S < \tau)$. Then $\alpha(g_\tau) + \beta(g_\tau) = F_A(\tau^-) + 1 - F_B(\tau^-)$, and the supremum over left limits equals the supremum over t by right-continuity, so

$$\inf_\tau \big(\alpha(g_\tau) + \beta(g_\tau)\big) \;=\; 1 - \sup_t \big(F_B(t) - F_A(t)\big) \;\geq\; 1 - \mathrm{KS}(P_S^A, P_S^B),$$

with $\mathrm{KS} \leq \mathrm{TV}$ because each $\{S \leq t\}$ is one admissible E in the supremum defining TV. The reversed family (open iff $S < \tau$) has $\inf_\tau = 1 - \sup_t(F_A(t) - F_B(t))$, so over half-lines of either orientation the tight constant is exactly $1 - \mathrm{KS}$, as claimed.

(iii) Claim: allowing randomization does not lower the floor of (i). A randomized rule is a measurable $\gamma : \mathbb{R} \to [0, 1]$, $\gamma(s) = P(\text{open} \mid S = s)$, with $\alpha(\gamma) = 1 - \int \gamma \, dP_S^A$ and $\beta(\gamma) = \int \gamma \, dP_S^B$. Then

$$\alpha(\gamma) + \beta(\gamma) \;=\; 1 - \int \gamma \,(p_A - p_B)\, d\mu \;\geq\; 1 - \int_{E^*} (p_A - p_B)\, d\mu \;=\; 1 - \mathrm{TV}(P_S^A, P_S^B),$$

since $0 \leq \gamma \leq 1$ and the integrand is maximized pointwise by $\gamma = \mathbb{1}[E^*]$. The floor of (i) therefore survives randomization, which proves the claim. ■

B EXPERIMENTAL SETUP AND THREATS TO VALIDITY

B.1 MODELS, IMPLEMENTATION, AND THE MASK FORMS

Table 6 details the models and the serving stack. Table 7 defines the two mask forms and their resulting candidate-set sizes. Finally, Table 8 summarizes the configuration labels used throughout §4.

component	configuration
host	Qwen2.5-7B-Instruct by default; 32B variant where noted
expert	Qwen2.5-0.5B-Instruct; one LoRA adapter per inventory, or a single adapter trained on the three API hubs for the request-level router
generality sweep	16 open-weight hosts (1B–72B) across four families (Qwen2.5 $\times$8, Llama-3.x $\times$4, Gemma-2 $\times$3, Mistral $\times$1), plus five frontier APIs
self-hosted coupling	custom logits processor in `transformers` (bf16) on A100, H100, or A40 GPUs
BFCL and BIRD	vLLM with xgrammar
closed API hosts	native structured-outputs features of the provider

Table 6: Models and implementation details.

mask	identifier constraint	HF	TF	Torch
flat	full inventory: one prefix tree over all K legal strings	895	80	94
anchored	children of a host-emitted legal anchor (e.g., domain or table name)	27	15	39

Table 7: The two mask forms and their resulting mean effective candidate-set size (K) at the identifier slot. While the anchored form shrinks K sharply, it relies on the host to generate the anchor. Consequently, under a hard mask, an anchor error propagates as a fatal failure rather than a recoverable one. We compare both forms empirically in §4.2.

label	configuration
host	frozen host alone
masked host	host + mask from §3.2
expert	bare expert decoding alone (serves as a reference, not a proposed method)
anti-expert	the base 0.5B model prior to tuning (used by the proxy, the KL gate, and the untuned-twin control)
blend(k, β)	host + β·expert logits over the candidate pool (*our proposed method*)
expert-decides / reset	blend($k, \beta{\to}\infty$): the expert exclusively ranks the host's k nominees (the deployed setting)
in-pool sum	blend(k, 1)
sum	host + expert logits over the full vocabulary (a product of experts; used as an ablation)
proxy / proxy-in-pool(k)	expert $-$ anti-expert, applied globally / inside the pool
masked expert / masked pair	bare expert, or host + expert at $\beta = 1$, decoding under the mask on the host's anchor
merged expert + mask / router-composed	merged expert decoding alone under the mask (no host pool), either always-on or toggled by the router's verdict
logit sum / proxy tuning	sum and proxy configurations from Table 5
single-inventory head	binary router head (TF Hub vs. all other domains)
gate($\cdot$) / router	arbiters from §3.4 applied atop any configuration. A gate evaluates a per-step statistic (raw confidence, KL divergence of the expert's learned change, or a learned head on the expert's state). The router evaluates the full prefill state.
serial	closed-API deployment (§4.3, T3; Algorithm 1)

Table 8: Configuration labels used throughout the experimental results in §4.

Success conditions. A method is deemed successful only if both conditions from §3.1 hold simultaneously **under a single, global setting**:

(A) In-inventory: performance must equal or exceed always-on baselines (logit sum, proxy tuning).
(B) Out-of-inventory: performance must match the host.

Requiring a per-benchmark setting constitutes a calibration failure and is reported as such. The deployed router meets both conditions on TF Hub, HuggingFace, TorchHub, and GSM8K (Table 5).

This evaluation protocol was pre-registered prior to analyzing the arbitration runs. All neural-sampler results are reported as the triplet of §3.1: accuracy on covered requests, accuracy on uncovered requests, and the bare host's baseline.

Cost. Table 9 shows the processing cost for a single request on an A100 GPU (our `transformers` prototype; the vLLM runs of BFCL and BIRD are not timed). When the mask is active, the system automatically forces many tokens without needing to run the model. Because of this, masked decoding is actually faster and cheaper than standard free-form generation. Adding the expert model roughly doubles the computation time, but only at the specific points where the model must choose between multiple legal options. In contrast, running both models for every single token costs 2.8 times as much as the base model alone; the deployed coupling, which has no mask, is in this class. Finally, the routing mechanism is highly efficient. It only adds one initial processing step by the small expert model for each request.

cell	ms per emitted token (TF / HF / Torch)	s per query
host, free-form	27.1 / 27.0 / 26.6	1.1–1.2
masked host	—	0.15–0.62
masked pair (β=1)	—	0.40–0.91
masked expert alone	—	0.28–0.74
host + expert, unscoped sum, no mask	74.9 / 74.2 / 73.6	3.2–3.4

Table 9: Decode cost per cell (Qwen2.5-7B host, 0.5B expert; batch 1, A100-40GB, 120-question slices per hub). Masked latency measurements cover only the prompt prefill and the identifier slot, excluding the host's domain line. When the mask restricts the output to a single legal continuation, the system forces the token without computing a forward pass. The entire forced span is then fed to each model in one pass at the next decision branch. Masked generation executes faster than free-form decoding both because it skips these forward passes and because it measures a shorter overall span. Per-token latencies are reported for unmasked configurations only.

B.2 The expert: recipe, regimes, and variants

Design rationale (for the expert of §3.3). We opt for a neural expert rather than a search index or rule system for three practical reasons:

1. it never comes up empty: it always ranks the available options, guaranteeing a legal output under the mask without needing messy fallback rules;
2. it is highly efficient, able to run directly inside the host model without wasting expensive prompt tokens;
3. it keeps the architecture clean by placing knowledge, masking, and routing at the exact same decision point.

Recipe. The expert is Qwen2.5-0.5B-Instruct with a LoRA adapter (r $= 16$, $\alpha = 32$, dropout 0.05, on the q/k/v/o/gate/up/down projections), trained with AdamW (lr 10^{-4}, weight decay 0.01, OneCycle with 6% warm-up), three epochs, token-budget batching, bf16, fixed seeds, and no early stopping. Training records combine card→identifier memorization lines, grammar-sampled inventory lines, question→gold-call pairs, and draft-repair pairs (a corrupted host draft mapped back to the gold). The target is the three-line gold output; for the per-inventory experts, the loss is token cross-entropy only on the identifier *value* spans (domain and call), with prompt and format tokens masked out.

The two data regimes. We evaluate our system under two data setups. The "log" setup uses the benchmark's training questions as a stand-in for real usage logs. In this setup, the model sees every target answer during training by design. The "inv" setup simulates a cold-start scenario with zero real user data. Here, the host model creates synthetic training questions directly from the API descriptions. We filter these synthetic questions so that none is an exact or near duplicate (Jaccard ≥ 0.7) of a test question. In both setups, training and testing data are kept separate, up to the residual overlap audited in Table 10.

What cold start actually buys. We tested the cold-start setup using zero real usage logs. Most of the performance gains actually come from the mask alone. The mask forces legal outputs without

hub	eval / train n	exact dup	Jaccard ≥ 0.7	≥ 0.7 same gold	gold key in train
TF Hub	688 / 6,190	0	9 (1.3%)	6 (0.9%)	100%
TorchHub	186 / 837	0	0	0	100%
HF	911 / 8,191	0	3 (0.3%)	3 (0.3%)	100%

Table 10: Train/eval overlap audit for the log-regime experts (lowercase-alphanumeric Jaccard over instructions parsed from both Gorilla record formats; raw Gorilla eval counts, of which the 21 questions whose gold has no parsable identifier are excluded from every accuracy table).

needing any training data. The cold-start expert adds only a small improvement ($+3.3$ over the flat mask; $+1.9$, n.s., over the anchored one). Furthermore, the routing mechanism fails without real logs. It struggles to recognize its own domain and mistakenly activates on foreign datasets. It does fail safely by staying turned off for general queries, but it misses many valid requests. The mask deploys without logs; in this experiment the router did not. Table 11 shows the details.

cell (inventory-only expert, no logs)	TF Hub (n=688)	HuggingFace (n=300)
host (free-form)	48.5	3.3
flat mask, host decides	54.4 (+5.8 [+3.9, +7.8])	2.7 (n.s.)
anchored mask, host decides	56.2 (+7.7 [+5.4, +10.2])	2.7 (n.s.)
masked expert alone	47.5 (n.s.)	5.3 (n.s.)
composed (flat mask + expert)	57.7 (+9.2 [+6.4, +12.1])	2.7 (n.s.)
composed (anchored mask + expert)	**58.1** (+9.6 [+6.7, +12.5])	3.0 (n.s.)
what each part contributes, paired against the cell it builds on		
the mask alone, over the host	$+5.8$ flat, $+7.7$ anchored	-0.7 (n.s.)
the expert, on top of the flat mask	$+3.3$ [$+1.0$, $+5.7$]	$+0.0$ (n.s.)
the expert, on top of the anchored mask	$+1.9$ [-0.1, $+4.1$] n.s.	$+0.3$ (n.s.)
the cold-start router: activation rate (%), by the head trained on that bed		
on its own catalog	28.2	33.7
on its worst foreign catalog	**32.8** (TorchHub)	**35.0** (ShadowHub)
on the arithmetic bed	0.0	0.0

Table 11: Cold start: the pipeline fitted from the inventory alone. Deltas are paired bootstrap against the same run's host cell; n.s. marks an interval containing zero. The middle block decomposes the gain, and the lower block gives the router's activation rates, whose log-regime reference on ShadowHub (the synthetic private catalog of the validity section below) is 97.0% own-bed at ROC-AUC 0.992; on TF Hub the log-regime head activates on 77.2% of its own requests (Table 30).

The merged expert. One adapter serves the three hubs: their log records (14,460 / 18,039 / 3,768 for TF/HF/Torch), each under its own serving prompt, with cross-hub negatives (29 / 35 / 20% of those counts; none between TF Hub and TorchHub, Appendix B.3) and 1,000 GSM8K-train questions, both labeled with the default (abstain) output. The router head's own default class is fitted on 3,000 GSM8K-train questions, the only general data it sees; Table 29 measures the consequence. Its per-class thresholds are tuned on a random 10% of the training records; because each question appears in several records, 28.7% of the development hub prompts repeat a question seen in training. The expert's prefill representation is the router head's input (§3.4): one sidecar, two functions.

Deployment pseudocode. The sidecar's two deployments, in-decode and serial, are given in Algorithm 1:

B.3 Threats to validity

The format example. Every number in the paper uses a placeholder in the prompt's three-line format example. Early runs that placed a real identifier there inflated the mask's gain (on the 7B, $+23.7$ on TF Hub against $+6.1$ with the placeholder), because hosts wrote variants of the example that the mask then converted.

Expert train/eval overlap. By construction of the log regime, every gold identifier appears in the expert's training data. This is the definition of the regime (the inventory owner holds usage logs) and is stated rather than hidden. At the *question* level, however, overlap is audited. No evaluation question is duplicated in the training set; near-duplicates (Jaccard ≥ 0.7) are $\leq 1.3\%$, and near-duplicates

Algorithm 1 Sidecar-augmented decoding. In the serial form for closed hosts, lines 1–2 execute locally on the sidecar before any external API call is made. During this step, only the expert e is prefilled. If an inventory class wins, the system calls the closed host once and passes the legal set via structured outputs. In this deployment, the expert's token-level knowledge cannot intervene. Only the constraint mask and the routing verdict travel to the host.

Require: request x; host h; sidecar (expert e, router head, legal sets $\{L_c\}$); pool size k, blend weight β, class thresholds $\{\theta_c\}$

1: prefill e on x
2: $c \leftarrow$ highest-scoring class with score $\geq \theta_c$; *default* if none {router, §3.4}
3: **if** $c =$ *default* **then**
4: decode with h on x; **return** {condition (B) path}
5: **end if**
6: prefill h on x
7: **while** decoding with h **do**
8: **if** the mask is on and not at an identifier slot **then**
9: emit h's token; **continue**
10: **end if**
11: $P \leftarrow h$'s top-k, intersected with L_c if the mask is on {order A or B, Appendix F}
12: **if** $P = \emptyset$ **then**
13: $P \leftarrow$ the legal continuations of L_c {fallback, §3.3}
14: **end if**
15: emit $\arg\max$ over P of $h(\cdot \mid \text{prefix}) + \beta\, e(\cdot \mid \text{prefix})$ {the blend, §3.3; h and e both consume the token}
16: **end while**

sharing the gold identifier are $\leq 0.9\%$ across every hub (TorchHub: zero). The audit finds little exact or high-Jaccard overlap, though it cannot exclude semantic overlap. Table 10 details the per-hub counts.

We applied the same audit to every host-generated training set, where questions are synthesized by the host rather than sampled from the benchmark. Here, we found no overlap at all. There are zero exact duplicates and zero near-duplicates (Jaccard ≥ 0.7) against the evaluation questions across all three hubs and ShadowHub. The closest single pair reached only 0.48 on TF Hub, 0.46 on TorchHub, 0.65 on HuggingFace, and 0.43 on ShadowHub.

TorchHub training prompts. Gorilla's TorchHub training split differs from its evaluation split and from the other hubs: it embeds each question inside its full source record, which includes the correct answer. We extract the clean question from each record before training; otherwise 805 of the 819 parsable records name the gold repository and model in the prompt. With clean questions, TorchHub and TF Hub requests overlap: TorchHub questions trained to abstain under the TF Hub prompt made the merged expert refuse TF Hub's own requests (no identifier on 88% of them), so the merged recipe keeps no negatives between these two catalogs. The single-inventory TF Hub and HuggingFace experts, the TorchHub expert of Table 13, the binary head, the ShadowHub and cold-start routing heads, the mixed-request probe (Appendix G), and the Falcon3 replication (Appendix F.4) were trained on the unrepaired records.

Official-checker alignment. The official Gorilla AST evaluator and our identifier match measure fundamentally different things. They diverge in both directions. First, the AST check is overly strict on surface forms. Minor deviations, such as a missing version digit or a paraphrased argument, count as hallucinations and zero out all free-form rows. Conversely, the AST check is too loose on identity. It accepts any structurally matched API within the gold target's *domain* as correct. This leniency credits masked outputs 30–80% of the time even when they name the wrong API. Consequently, masked generation looks artificially spectacular under the official metric. We adopt our identifier metric as the more conservative choice for evaluating mask claims. Because per-question agreement between the two metrics varies widely (43–97% per cell), published Gorilla numbers cannot be directly compared to our baselines without metric alignment.

Nonce inventories: demonstration, not leakage. We renamed every TF Hub identifier to a consistent nonce across the catalog, prompt, and gold target. This intervention collapses the flat-mask cell from 53.2% to 2.2% and reduces the anchored cell to template-level chance (26.3%). This demonstrates that the mask's gain depends on the host's latent knowledge of the real names, precisely

matching our support-recall hypothesis. Conversely, it shows the mask itself does not leak the correct answer.

A genuinely private catalog (ShadowHub). To test our system on entirely unseen data, we created ShadowHub. This is a private catalog of 200 fabricated identifiers across 20 domains. Because these names are invented, the base model should not be able to guess them on its own. We observed three main results. First, without any usage logs, the system can still restrict outputs to correct, legal domains 65.5% of the time (the anchored mask alone: 63.0%). However, it cannot reliably pick the exact item within a domain because the items share similar descriptions. Second, when trained on synthetic usage logs (on a rebuilt bed where each API carries a persistent use case), our masked expert model successfully retrieves the exact private identifier. It achieves 11.5% accuracy on unseen phrasing, which is 23 times better than random chance and more than twice the best mask alone (5.0%). Finally, a routing head trained on the log-regime ShadowHub expert learns when to activate it. It correctly triggers on 97.0% of ShadowHub requests, safely ignores unrelated tasks such as math questions, and misfires on 1.2% to 13.4% of the real hubs' requests, TorchHub being the highest. The head learns when its inventory applies, on a catalog the host never saw.

Paraphrased-eval probe. We tested whether the expert's performance stems from memorizing the benchmark's phrasing by having the host paraphrase each TF Hub evaluation instruction. We enforced a strict no-surface-forms constraint, ensuring no model names, URLs, or API strings appeared in the new prompts. Any leaky paraphrase would have reverted to the original string and been counted, but none occurred. Rerunning the factorial on this paraphrased dataset yielded no performance changes outside statistical noise (Table 12). Our pre-registered falsifier (that the expert's performance edge would collapse toward the cold-start floor if it relied on phrasing memorization) did not occur. This shows the expert's gain is robust to how a request is worded, arguing against phrasing memorization. Because the paraphraser failed on a malformed record partway through, the paraphrased dataset consists of the first 428 of the 688 evaluation questions.

cell	paraphrased bed	original bed
host	49.5	50.0
bare expert	68.9	69.6
the expert's edge over the host	$+19.4$	$+19.6$
flat mask (Δ vs host)	$+5.6$	$+5.4$
reset $\cap$ pool-50 ($n = 300$)	68.0	69.3

Table 12: The paraphrased-eval probe: every cell of the TF Hub factorial re-run on paraphrased instructions, against the same cell on the same questions' original instructions (the first 428; the pool-50 row, the first 300).

Protocol variation across tables. Baseline scores differ slightly across tables because the evaluation protocols differ. For instance, the host's free-form accuracy on TF Hub is reported as 49.6 under the 688-question harness, 49.1 under the factorial protocol (n=688; 49.0 in the merged expert's rerun), and 48.5 on the 200-question subset behind the router tables (48.0 in the rerun behind the regret table) and on the lean full-bed reruns behind Tables 11 and 18. Similarly, expert configurations yield 67.4 (masked merged expert, n=688), 70.0 (unmasked reset coupling, n=200), and 69.8 (ungated expert-decides at k=200). Every table specifies its active protocol, and the released ledger maps all printed numbers to their raw record files.

C Full grids: gates, pool size k, and candidate-set size K

Raw-confidence gate. Under this gate, the expert decides the next token whenever its own top probability reaches τ. Otherwise, the host decides (Table 13).

KL-gate thresholds across three conditions. Under the KL gate, the expert decides the next token whenever its KL divergence from the anti-expert, over the host's top 50 tokens, reaches κ (Table 14).

Which steps Figure 3 counts, and the slot-only view. Figure 3 evaluates all decoding steps, including basic formatting. We reran the measurement with token positions recorded (Table 15) to focus only on the steps that generate the actual identifier. Even in this restricted view, the signals

token-level gate(τ)	TF Hub (strong expert)	HF (weak expert)	TorchHub (weak expert)
$\tau = 0.5$	**70.7** (halluc 0.0%)	**7.3** (21.0%)	4.4 (11.5%)
$\tau = 0.8$	66.3 (3.0%)	2.3 (33.0%)	7.1
$\tau = 0.95$	61.0 (10.3%)	2.7 (42.3%)	= host
host, same questions	48.3	3.3	7.7
ungated reset, full n	68.8	1.6	1.6

Table 13: Raw-confidence gate, token-level ($n = 300$ on TF Hub and HuggingFace, 183 on TorchHub). Each column uses that dataset's own expert.

KL gate (token-level ∩ pool50)	A: TF expert on TF questions	B: TF expert on HF questions	C: HF expert on HF questions (weak)
$\kappa = 0.3$ nats	64.3 (halluc 6.7%)	0.0 (−3.3; 78% foreign names)	4.7
$\kappa = 0.8$	58.3	1.3 (−2.0; 79% foreign names)	—
$\kappa = 1.0$	57.0 (18.0%)	—	4.4
$\kappa = 1.5$ / 2.5 / 4.0 (per token)	55.0 / 53.0 / 51.0	2.0 / 3.0 / 3.0 (≈ host from 2.5)	— / — / 4.9
$\kappa \geq 1.5$ (per slot)	48.5 = host (never opens)	2.0–2.5	—
references	full-vocabulary sum 66.3 and proxy 64.3 (n=300); reset 68.8 (n=688)	host 3.7 (3.3 @300, 2.5 @200)	host 3.7, ungated masked expert (host's anchor) 5.5

Table 14: KL-gate grid. Condition C was rerun in full (masked, $n = 893$). In A and C, no threshold reaches the ungated reference. A dash marks an unmeasured cell. In column A, n is 300 for $\kappa = 0.3$ and 1.0, 688 for $\kappa = 0.8$, and 200 for $\kappa \geq 1.5$. Column B uses 300 below $\kappa = 1.5$ and 200 from 1.5.

from different datasets overlap heavily. This confirms the overlap is not merely an artifact of shared formatting tokens. The only token that clearly separates the datasets is the first identifier token. However, the token reflects the prompt's template rather than the actual query, and the signal's sign is reversed: the expert alters the distribution far more on the foreign dataset where it must stay closed. This explains why the per-slot gate of Table 14 never opens on the expert's native catalog. The host's first identifier token is the serving template's own token (`hub` on every TF Hub request, `pipeline` on 299 of 300 HuggingFace requests). The TF expert's learned change on this token is 0.01 nats under its own template, compared to 1.23 nats under the foreign one. Ultimately, averaging these token-level signals across the entire request narrows the overlap but still leaves an error floor of 0.38 to 0.63.

	must-open (TF Hub)		must-close (HuggingFace)			
statistic S	median	p_{90}	median	p_{90}	TV	floor $1 - \mathrm{TV}$
all 64 decode steps (Figure 3)	0.017	0.68	0.022	1.09	0.05	0.95
steps before EOS	0.011	1.01	0.024	1.80	0.08	0.92
identifier-slot steps	0.049	0.94	0.119	1.60	0.12	0.88
identifier-slot steps after the first	0.054	0.96	0.094	1.41	0.09	0.91
first identifier token (one per request)	0.011	0.09	1.228	4.43	0.87	0.13
request mean over all 64 steps	0.286	0.46	0.368	0.49	0.37	0.63
request mean over the slot steps	0.278	0.63	0.526	0.81	0.62	0.38
request mean over the slot steps after the first	0.289	0.65	0.441	0.69	0.45	0.55
request maximum over the slot steps after the first	1.906	4.49	2.368	4.19	0.38	0.62

Table 15: The learned-change KL of Figure 3, rerun with token positions recorded. Host, expert, datasets, and the 64-step budget are unchanged, with 300 requests per dataset. Values are in nats, and TV uses 0.05-nat bins. The rerun lets the host emit EOS, and the steps after it sit near zero. The decode of Figure 3 was forced to 64 tokens. The rerun's medians are therefore lower than the main text's 0.041 and 0.045, while the TV matches.

k-curves: the pool helps retrieval but does not ensure safety. The pool size k is the number of the host's top tokens among which the expert chooses (Table 16). With the strong TF Hub expert, accuracy rises with k and hallucination mostly falls. With the weak HuggingFace and TorchHub experts, a larger pool does not help. Their accuracy stays below the host at every k.

BIRD, pre-registered. Before the BIRD run, we registered a prediction: with the schema in the prompt, the mask would add 0 to 2 points and remove invented names. The result reported in §4.2 came close but fell just outside it.

token-level expert-decides	$k = 5$	$k = 10$	$k = 25$	$k = 50$	$k = 200$
TF, log expert (strong)	65.0 (5.7%)	67.0 (2.3%)	67.7 (0.7%)	69.3 (0.0%)	69.8 (0.3%)
TF, inv expert (weak)	—	52.0	—	56.0	—
HF, log expert (weak)	3.0 (56%)	2.0 (58%)	2.7 (45%)	1.6 (36%)	—
TorchHub, log expert (weak)	5.5 (16%)	6.6 (9%)	4.9 (10%)	4.4	—

Table 16: k-curves, with the hallucination rate in parentheses. TF Hub and HuggingFace cells use 300 questions (host 48.0 to 48.3 and 3.3). The exceptions are TF Hub at k=200 (688, host 49.1) and HuggingFace at k=50 (893, host 3.7). TorchHub uses all 183 (host 7.7). For the weak inv expert, in-pool *voting* (not shown) reaches 64.0 to 64.3, 8 to 12 points above expert-decides. So who decides should depend on the expert's competence.

BFCL controlled-K grid. Here K is the candidate-set size: the gold target plus $K - 1$ distractors. It is not the pool size k of the curves above. On the 7B host with schemas in the prompt (P1), the name mask raises full-AST accuracy by 15 to 32 points at every K. Invented names drop to zero (Table 17). Without schemas (P2), legality alone lifts name accuracy from 12.0 to 61.0 at K=1, and abstention stays at 86 to 88% (85.9% without the mask). The frontier models almost never invent a name when schemas are present ($\leq 0.6\%$), so the mask has nothing to fix. Instead, it *costs* them about 16 points of irrelevance accuracy at every K.

7B, P1 (schemas in prompt)	$K = 1$	$K = 5$	$K = 10$	$K = 20$	$K = 50$
free-form full-AST	37.7	24.2	20.2	13.1	8.6
name-mask full-AST	52.5	42.4	41.3	41.2	40.5
name-mask name acc	92.9	74.0	71.7	69.9	68.8

Table 17: BFCL controlled-K grid (hard distractors, irrelevance split included). The frontier rows and the abstention-cost columns are in the released records.

D FULL TABLES: BRACKETS, TWINS, AND SECONDARY ROWS

A value marked † is recorded in the experiment ledger but its per-question records were not retained (the two systematic losses are k-sweep runs that overwrote one output file per k, and legs that predate the surviving backups); such cells carry no bracket anywhere in the paper.

Mechanism, and the measured bound. Property 1 gives the mask's slot-level gain in the value-level idealization: $P(\hat{y} \notin L \wedge \hat{y}_L = \text{gold})$. Table 1 displays this bound alongside the realized gain. The bound is computed from free-form records and measured legal rankings; no masked outcome enters its definition. Our realized gains never exceed it. On TF Hub, the anchored mask converts the entire 6.1-point ceiling, while the flat form captures $+3.6$ points. On TorchHub and HuggingFace, the ceilings are zero and 0.4. The negative gains there (-2.7 and -1.1 for the 7B host) are therefore correct-to-wrong flips. Under greedy decoding, a prefix-trie mask that continues the host's token stream cannot alter an output whose tokens already follow a catalog key's path. Because our masked decoding re-encodes the prefix, this holds approximately. On HuggingFace, the mask therefore mostly alters outputs that the identifier metric credits but that are not catalog keys. On the 7B host's factorial run, 21 of the 26 HuggingFace flips have a gold key written in a call form other than the prompt template's `pipeline(...)` (for example, `DPTForDepthEstimation.from_pretrained`). The other five are translation questions whose gold key names a language-pair task. There the host writes the generic `translation` task, and the trie then allows only the one key with that task. On TF Hub, five of the eight anchored-mask flips come from host anchor errors. The mask enforces exact keys, while the metric checks only the identifier. Most HuggingFace losses reflect this gap rather than token-level distortion. Finally, the bound grows with host scale. Larger hosts produce more illegal errors that are near-misses of the gold (36.8% at 32B against 2.6% at 7B), which gives the mask more to convert.

Twin of Table 1: paired deltas of the rules family. Table 18.

The full host grid behind Table 2. Table 19 gives the full grid.

Δ vs host [95% CI]	TF Hub	TorchHub	HF	BIRD (EX)	CypherBench (EX, held-out)	BFCL (full-AST, K=10)
flat mask	+3.6 [+1.2, +6.1]	−1.6 [−4.4, +0.5]	−1.2 [−2.6, +0.2]	+2.5 [+0.6, +4.2]	n/a	+21.0 [+19.3, +22.8]
anchored mask	+6.1 [+3.3, +8.9]	−2.7 [−6.6, +0.5]	−1.1 [−2.4, +0.1]	n/a	+2.1 [+1.4, +2.8]	†
masked expert (host's anchor)	+9.7 [+6.2, +13.1]	−3.8 [−8.2, +1.1]	+1.8 [+0.1, +3.7]	n/a	n/a	n/a
masked pair (flat / anch)	+12.4 [+9.2, +15.4] / +12.8 [+9.6, +15.8]	−1.1 [−3.3, +1.1]	†	n/a	n/a	n/a

Table 18: Paired brackets behind Table 1 (Gorilla n = 688/183/893; BIRD 1,534; BFCL 2,150; CypherBench 2,048 held-out, paired on retained records). Rows pair against their own run's host: 49.6 / 8.2 / 3.5 for the mask rows, 49.1 / 7.7 / 3.7 for the masked expert, 48.5 / 7.7 for the masked pair (Appendix B.3).

host	TF free	TF masked (Δ, form)	HF free	HF masked (Δ)	SR@1 TF	HF	Torch
Qwen2.5-1.5B	34.0	**40.7** (+6.7, anch)	2.1	1.2 (−0.9)	38.2	0.7	1.6
Qwen2.5-3B	43.3	43.3 (+0.0 n.s., anch)	2.2	1.6 (−0.7)	41.4	0.9	3.8
Qwen2.5-7B	49.6	**55.7** (+6.1, anch)	3.5	2.4 (−1.1)	51.9	1.8	7.7
Qwen2.5-Coder-7B	43.0	**52.6** (+9.6, anch)	4.1	**2.1** (−2.0)	43.9	2.9	5.5
Qwen2.5-14B	51.9	**62.9** (+11.0, anch)	4.9	**2.9** (−2.0)	62.4	2.2	7.7
Qwen2.5-32B-AWQ	34.4	**50.9** (+16.4, anch)	5.3	**3.1** (−2.1)	43.3	3.2	9.3
Qwen2.5-32B	33.3	**52.0** (+18.8, flat)	5.8	**3.5** (−2.4)	44.6	3.8	5.5
Qwen2.5-72B-AWQ	47.8	**51.7** (+3.9)	4.6	3.2 (−1.3)	59.0	4.1	7.7
Llama-3.2-1B	0.3	**16.7** (+16.4, anch)	0.3	0.9 (+0.6 n.s.)	30.2	0.1	1.6
Llama-3.2-3B	0.3	**22.5** (+22.2, anch)	0.8	1.0 (+0.2 n.s.)	55.2	0.0	2.7
Llama-3.1-8B	27.6	**37.9** (+10.3, anch)	1.9	1.8 (−0.1 n.s.)	39.7	0.8	2.2
Llama-3.3-70B-AWQ	35.2	**46.8** (+11.6, anch)	2.4	2.5 (+0.1)	—	—	4.9
Gemma-2-2B	21.1	**41.9** (+20.8, flat)	1.7	2.0 (+0.3)	36.2	1.0	2.2
Gemma-2-9B	32.6	**49.0** (+16.4, anch)	2.8	1.7 (−1.1 n.s.)	45.2	2.0	4.9
Gemma-2-27B	34.3	**51.9** (+17.6, anch)	3.8	**2.0** (−1.8)	44.8	1.2	4.9
Mistral-7B-v0.3	19.9	**40.6** (+20.6, anch)	1.6	1.6 (+0.0 n.s.)	46.9	1.2	3.3
gpt-4o-mini	35.5	**59.4** (+24.0)	4.4	**6.4** (+2.0)	—	—	—
gpt-4o	46.4	**61.8** (+15.4, flat)	6.8	**14.7** (+7.8)	—	—	—
gpt-4.1-mini	59.0	62.1 (+3.1, flat)	5.9	**12.7** (+6.7)	—	—	—
gpt-4.1	40.7	**64.1** (+23.4, flat)	7.4	**17.0** (+9.6)	—	—	—
o4-mini (low)	50.4	**62.4** (+11.9)	7.4	**12.0** (+4.6)	—	—	—

Table 19: The full 21-host grid behind Table 2: identifier-match accuracy (%), free-form vs constrained decoding (best mask form; Δ in parentheses). TF Hub is the catalog the hosts partially know (K=80), HuggingFace the one the open hosts do not (K=895). Bold: the delta's 95% paired-bootstrap CI excludes zero and its bootstrap $p < 0.05$ (see the multiple-comparisons paragraph). TorchHub column in the appendix twin. SR@1: support recall at rank 1 over the legal set, the probe of Figure 2 (open hosts; the 70B was probed on TorchHub only).

Provenance of the support-recall grid (Figure 2). We measured Support Recall (SR) using log-probabilities over the legal set, utilizing each host's own domain line as the prefix under the placeholder-example protocol. In total, we tested 46 model-catalog pairs across every open host: 15 on TF Hub, 15 on HuggingFace, and 16 on TorchHub. We probed Llama, Gemma, and Mistral checkpoints through weight-identical ungated mirrors. To manage high computing costs, we evaluated the 70B model exclusively on TorchHub. We also tested three quantized models in a separate software environment to avoid dependency conflicts. While the quantized 32B model scored within 3.8 SR@1 points of its full-precision twin on every dataset, we excluded all quantized models from the correlation analysis in Figure 2b to prevent mixed weight precisions from skewing the results. Ultimately, we found the SR metric to be highly stable. While changing the prompt shifted standard accuracy by 15 points, it moved SR@1 by less than 3 points. Because closed APIs cannot score arbitrary candidates, we can only calculate SR directly for open models, though closed models follow the same performance trends. Finally, Figure 5 shows the full distribution of these rankings across the different datasets.

The TorchHub column and full brackets. Table 20 extends Table 2 by reporting the best-form masked delta per host alongside their full confidence intervals. The TorchHub column is where the boundary shows: its deltas fall on both sides of zero, and the two largest losses anywhere in the grid (Qwen2.5-32B -4.9 and its AWQ twin -4.4) sit there.

Multiple-comparisons control. The pooled Holm family contains 73 of the paper's bracketed deltas (the 63 cells of this grid, the two masked-pair cells, and eight headline claims), corrected at $\alpha = 0.05$; the other brackets are uncorrected. Every core claim survives it:

- **Survives:** all TF Hub grid gains of $+6$ or more; the masked pair; the frontier API gains on TF Hub and HuggingFace (gpt-4.1-mini on TF Hub and gpt-4o-mini on HuggingFace excepted) and

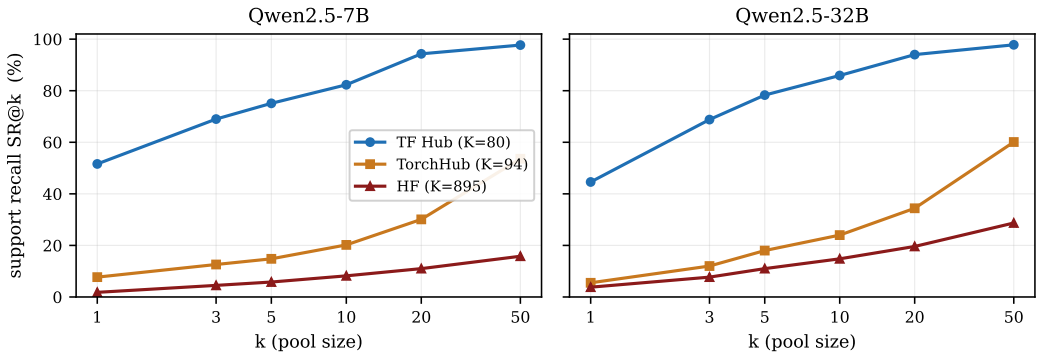

Figure 5: Support recall SR@k over the legal set, 7B and 32B hosts × three catalogs (placeholder-prompt protocol); the full-grid view is Figure 2.

host	TF Hub Δ [CI]	HF Δ [CI]	TorchHub Δ [CI]
Qwen2.5-1.5B	+6.7 [+4.1, +9.3]	−0.9 [−2.0, +0.2]	+2.2 [+0.5, +4.4]
Qwen2.5-3B	+0.0 [−2.8, +2.9]	−0.7 [−1.8, +0.4]	+3.3 [+1.1, +6.0]
Llama-3.2-1B	+16.4 [+13.8, +19.2]	+0.6 [+0.0, +1.2]	+2.2 [+0.5, +4.4]
Gemma-2-2B	+20.8 [+17.7, +23.8]	+0.3 [−0.7, +1.3]	+2.7 [+0.5, +5.5]
Qwen2.5-7B	+6.1 [+3.3, +8.9]	−1.1 [−2.4, +0.1]	−1.6 [−4.4, +0.5]
Qwen2.5-Coder-7B	+9.6 [+7.3, +12.2]	−2.0 [−3.5, −0.7]	+1.1 [+0.0, +2.7]
Qwen2.5-14B	+11.0 [+8.3, +13.8]	−2.0 [−3.5, −0.6]	+0.0 [−2.2, +2.2]
Qwen2.5-32B-AWQ	+16.4 [+13.7, +19.5]	−2.1 [−3.9, −0.4]	−4.4 [−7.7, −1.1]
Qwen2.5-32B	+18.8 [+15.8, +21.9]	−2.4 [−4.0, −0.7]	−4.9 [−8.2, −2.2]
Qwen2.5-72B-AWQ	+3.9 [+1.3, +6.7]	−1.3 [−3.0, +0.3]	+3.8 [+1.6, +6.6]
Llama-3.2-3B	+22.2 [+19.3, +25.3]	+0.2 [−0.6, +1.1]	+3.8 [+1.1, +7.1]
Llama-3.1-8B	+10.3 [+7.6, +13.1]	−0.1 [−1.2, +1.0]	+3.3 [+0.5, +6.6]
Llama-3.3-70B-AWQ	+11.6 [+8.7, +15.0]	+0.1 [−1.1, +1.3]	+5.5 [+2.7, +8.7]
Gemma-2-9B	+16.4 [+13.7, +19.3]	−1.1 [−2.4, +0.1]	+3.8 [+1.1, +6.6]
Gemma-2-27B	+17.6 [+14.2, +21.1]	−1.8 [−3.4, −0.3]	+3.8 [+1.1, +7.1]
Mistral-7B-v0.3	+20.6 [+17.6, +24.1]	+0.0 [−1.1, +1.1]	+2.2 [−1.1, +6.0]
gpt-4o-mini	+24.0 [+20.5, +27.6]	+2.0 [+0.2, +3.8]	+7.1 [+2.7, +12.0]
gpt-4o	+15.4 [+12.2, +18.5]	+7.8 [+5.3, +10.3]	+3.8 [−0.5, +8.2]
gpt-4.1-mini	+3.1 [+0.1, +6.1]	+6.7 [+4.1, +9.2]	+6.0 [+2.7, +9.8]
gpt-4.1	+23.4 [+19.9, +26.9]	+9.6 [+6.9, +12.1]	+7.7 [+3.3, +12.6]
o4-mini (low)	+11.9 [+8.7, +15.4]	+4.6 [+1.9, +7.2]	+6.0 [+1.6, +10.9]

Table 20: Brackets behind Table 2; every cell is paired on retained per-question records. The four smallest hosts, the HuggingFace and TorchHub columns of six mid-size hosts, and the frontier TorchHub column were re-run to regenerate lost records; re-measured frontier cells differ from the originals by at most 1.5 points, as API snapshots move.

gpt-4.1's and gpt-4.1-mini's on TorchHub; Llama-3.3-70B's TorchHub gain; the CypherBench held-out gain; the always-on coupling; the router's full-bed cost; the Falcon3 replication.

- **Does not survive:** the smaller shifts ($|\Delta| \leq 7.7$), 23 of the bold-eligible cells: most TorchHub gains (Llama-3.3-70B's, gpt-4.1's, and gpt-4.1-mini's survive), the mid-size hosts' HuggingFace losses and the two 32B TorchHub losses, Qwen2.5-72B-AWQ on TF Hub, the two frontier cells above, and the two decision tests below, which pass within their own family. We report the grid cells as exploratory.

The pre-registered decision tests form a distinct family of seven; within that family Holm passes every bracketed claim, including the k=200 gated gap (+3.0, $p = 0.0033$) and the learned gate's on-catalog shortfall (−4.7, $p = 0.013$). Bootstrap p-value estimates ($B = 100{,}000$) recomputed from the retained records flag a single cell: gpt-4.1-mini on TF Hub (+3.1 [+0.1, +6.1]; at $B = 100{,}000$ the bracket is [+0.0, +6.1]) gives a two-sided $p = 0.058$, so its delta is printed unbolded.

Twin of Table 5: brackets for the finalized cells. Table 21 provides the paired brackets. We regenerated all off-bed cells that have retained per-question records. Sample sizes (n) vary across

the always-on, router, and learned-gate rows, with TF cells paired directly against the host baseline for the same run. Off-bed cells are identical across all multiclass router rows because a TF strategy on a foreign request scores zero, whereas a default verdict safely returns the host. For the logit-sum row, the TF cell uses the in-pool sum (k=200), while off-bed cells use the full-vocabulary sum (Appendix B). The router's specific zeros reflect different behaviors. It defaults perfectly on GSM8K, making its zero exact. On HuggingFace, the zero is measured: the router misrouted 1.2% of requests to TF, but the host had already answered these incorrectly. On TorchHub, the router sends no request to TF. The learned gate's TorchHub cell acts as a floor, since the base host scores 0.0 under that gate's foreign prompt. Furthermore, these router rows deploy only the TF Hub strategy, meaning foreign misroutes safely return the host. Table 3 arms all inventories, so misroutes trigger a foreign strategy and score zero. The TorchHub difference ($+3.8$ in Table 3 versus $+0.0$ here) comes from Table 3 applying the TorchHub strategy to the TorchHub requests routed to their own class. Finally, prompt framing alters these fallback scores. On the three API datasets, each request retains its serving prompt; a default or foreign verdict returns the host's answer under that prompt, as deployed. If the app instead strips the frame on a default verdict (Appendix F), such requests are answered bare. Decoded bare, the host names the correct TF Hub identifier in only 1% of all 688 answers and in one of the 157 answers the router returns to it. Consequently, under the stripping variant, the TF Hub cell drops to 57.4 ($+8.3$ $[+4.5, +11.9]$) rather than 64.8.

method (setting)	TF own	HF questions	Torch questions	GSM8K	n (TF / HF / Torch / GSM8K)
expert-decides always-on (k=200 on TF Hub, 50 elsewhere)	+20.6 [+17.4, +23.8]	−3.3 [−5.3, −1.3]	−7.7 [−12.0, −3.8]	−32.0 [−39.5, −24.0]	688 / 300 / 183 / 200
logit sum (in-pool on TF Hub; full-vocabulary off-bed)	+14.7 [+11.8, +17.6]	−1.3 [−2.7, −0.3]	−3.3 [−7.1, +0.5]	−10.5 [−16.5, −4.0]	688 / 300 / 183 / 200
proxy tuning	—	−2.0 [−3.7, −0.7]	−2.2 [−4.9, +0.5]	−1.5 [−6.5, +3.0]	688 / 300 / 183 / 200
proxy-in-pool (κ=0, k=25)	+15.0 [+10.3, +19.7]	−2.0 [−3.7, −0.7]	−2.2 [−4.9, +0.0]	+0.5 [−3.5, +4.5]	688 / 300 / 183 / 200
raw-conf gate (τ=0.8)	+16.4 [+13.5, +19.5]	−1.3 [−3.0, +0.3]	−1.6 [−4.4, +1.1]	−6.0 [−12.0, +0.0]	688 / 300 / 183 / 200
KL-gated expert-decides (0.8 ∩ pool50)	+9.7 [+6.7, +12.5]	−2.0 [−4.0, +0.0]	−6.0 [−9.8, −2.7]	—	688 / 300 / 183 / 200
learned per-token gate (head on the expert's state)	+14.7 [+8.7, +20.7]	−1.7 [−3.1, −0.5]	0.0 (floor)	+0.0 [+0.0, +0.0]	150 / 418 / 91 / 100
router → coupling, deployed (expert-decides, host top-200, n=688)	+15.7 [+12.8, +18.5]	+0.0 [+0.0, +0.0]	+0.0 [+0.0, +0.0]	exact 0	688 / 893 / 183 / 200
router → coupling (k=50, n=200)	+19.0 [+13.5, +24.5]	+0.0 [+0.0, +0.0]	+0.0 [+0.0, +0.0]	exact 0	200 / 893 / 183 / 200
router → bare expert (no host; n=688 / n=200)	+14.2 [+11.2, +17.3] +15.0 [+9.0, +21.0]	+0.0	+0.0	exact 0	688 (200) / 893 / 183 / 200
router → TF expert alone under the flat mask (n=688)	+8.3 [+5.4, +11.2]	+0.0	+0.0	exact 0	688 / 893 / 183 / 200
router → merged expert alone under the flat mask (n=688; its TF cell is Table 3's composed cell)	+15.3 [+12.4, +18.2]	+0.0	+0.0	exact 0	688 / 893 / 183 / 200
single-inventory head	+18.2 [+15.1, +21.1]	—	−3.8	—	

Table 21: Paired brackets behind Table 5; a dash marks a cell with no retained paired record, the last column gives each row's n per bed, and the surrounding text gives each row's protocol.

One sampling setting. All main results in the paper use greedy decoding. Table 22 tests our system using nucleus sampling instead. The overall performance trends remain consistent. Our proposed method still provides large accuracy gains over the base model (+21.3 points). It also keeps the hallucination rate extremely low (0.7%), whereas the sampled base model hallucinates frequently (34%). While sampling reduces accuracy in most setups, by 0 to 11 points, the small expert model suffers the most when acting alone. However, keeping the host model involved limits this drop to at most 4 points. Note that these specific results are based on a single random seed and dataset configuration; the intervals reflect question sampling, not seed variability.

Parity and the training recipe (the matched-records monolith). Table 23.

E ACTIVATION ACCOUNTING AND THE REGRET TABLE

The accounting relations. We establish two accounting relations to predict system accuracy on any dataset b. These rely on tuning each class's threshold on a development split, assigning a default

cell (TF Hub, $n = 300$)	greedy	sampled	sampled − greedy [CI]	sampled: Δ vs host [CI]	halluc. % (sampled)
host, free-form	48.3	44.7	−3.7 [−7.0, +0.0]	—	34.3
bare expert (0.5B), free-form	69.0	57.7	−11.3 [−16.0, −6.7]	+13.0 [+7.0, +19.3]	0.0
anchored mask, host	58.0	56.7	−1.3 [−4.3, +1.7]	+12.0 [+7.3, +16.7]	0.0
flat mask, in-pool sum ($\beta = 1$)	63.3	63.7	+0.3 [−1.7, +2.3]	+19.0 [+14.3, +23.7]	0.0
flat mask, expert alone decides	61.0	53.0	−8.0 [−12.3, −3.3]	+8.3 [+2.7, +14.0]	0.0
expert-decides, host top-200, no mask (the coupling)	70.0	66.0	−4.0 [−7.7, −0.3]	+21.3 [+16.0, +27.0]	0.7

Table 22: One sampling setting ($T = 0.7$, top-p 0.9, seed 0, one draw per cell) against the greedy run on the same 300 TF Hub questions, 7B host + TF log expert; every choice the cells make (the free-form token, the in-pool choice, the trie branch) is drawn from the same score vector (at an expert-decides step, the expert's tempered distribution over the host's pool). Paired bootstrap brackets, 2,000 resamples.

configuration (matched training records)	TF Hub	n
LoRA-tuned 7B host, log records	70.2	688
LoRA-tuned 32B host, log records	69.6	688
LoRA-tuned 32B host, inv records	70.3†	688
masked 0.5B sidecar (own anchor)	70.2	688
token PoE on the tuned 7B host	69.6 vs 70.2: −0.6 [−1.7, +0.6] n.s.	688
token PoE on the tuned 32B host	70.8 vs 70.3: +0.4 [−0.4, +1.3] n.s.	688

Table 23: The parity block. On matched records, the masked 0.5B sidecar achieves parity with LoRA-tuning the base host. This is an advantage, as our served-host setting explicitly prohibits the weight access required for host tuning. Adding the expert to an already-tuned host yields no further gains, indicating the tuned host already holds the expert's knowledge. Finally, supervising the identifier alone reaches only 5.5 on the least-known catalog (Table 1), against 17.9 for the merged expert trained on the full target; the loss and the merging are confounded.

class to the host, and assuming a strategy's per-request gain is independent of the router's verdict:

$$\text{(loss)} \quad A_b \approx A_b^{\text{host}} - \textstyle\sum_c \alpha_{c,b}\,\delta_{c,b}, \tag{1}$$

$$\text{(capture)} \quad A_b \approx A_b^{\text{host}} + \text{recall}_c\,(A_b^{\text{strat}} - A_b^{\text{host}}) \quad \text{on class } c\text{'s own bed.} \tag{2}$$

Here, $\alpha_{c,b}$ represents class c's activation rate on dataset b, and $\delta_{c,b}$ represents the damage of incorrectly applying strategy c. These parameters are measured directly from the data, so each relation holds only on the datasets where its activation rate was measured. Activation rates are reported in Tables 25, 35, and 29, while damages correspond to the always-on losses in Table 5. These relations are accounting approximations rather than strict bounds, as the independence assumption is not perfectly exact. For example, on the full TF Hub dataset, the capture relation predicts an accuracy of 65.1, while the measured accuracy is 64.8. This discrepancy occurs because the requests the router incorrectly defaults to the host are on average ones where the expert coupling would have yielded above-average gains. Routing errors vary by dataset. On math questions, the router perfectly defaults to the base model, yielding a zero error rate. HuggingFace requests also behave within expected limits. However, the router makes frequent mistakes on unfamiliar general tasks (Table 29). On the highly similar sibling catalog, it never triggers the TF Hub expert under its own prompt. Finally, on the expert's own catalog, the router misses 23% of valid queries. The capture relation estimates this recall gap at 4.5 to 4.7 points, which aligns with our measured shortfalls of 3.0 to 4.9 points. This represents the standard selective-classification trade-off (Geifman & El-Yaniv, 2017), applied here to a sampler strategy rather than a bare prediction.

The granularity margin. On the same suite, the best scalar gate at one setting (raw confidence, $\tau = 0.8$) leaves 13.2 points of regret; the router leaves 3.0, all on its own catalog.

rule (one setting across beds)	regret	PGR	$A(1)$	$A(0)$
host alone	22.0	0%	48.0	67.0
always-on expert-decides	43.4	< 0	70.0	35.0
best raw-confidence threshold ($\tau = 0.8$)	**13.2**	40%	65.7	61.0
KL gate, one setting (κ=0.3, masked, pool 50)	18.6	15%	56.8	65.5
strict KL ($\kappa = 4.0$)	19.0	14%	51.0	—
learned per-token gate (head on the expert's state) its own suite, n = 150 / 418 / 91 / 100	6.4	67%	63.3	= host
per-request router (multiclass + default)	**3.0**	**86%**	**67.0**	**67.0**
per-bed oracle (= a rule reading the frame, on this suite)	0	100%	70.0	67.0

Table 24: Regret against the per-bed oracle (summed over the four-bed suite: TF Hub, HuggingFace, TorchHub, and GSM8K), oracle-versus-host gap recovered (PGR), and deployed-accuracy endpoints. Deployed accuracy $A(p)$ equals $p\,A(1) + (1-p)\,A(0)$, where p is the in-inventory fraction, $A(1)$ is TF Hub accuracy, and $A(0)$ is GSM8K accuracy. On this stream, the always-on configuration overtakes the router only when $p \gtrsim 0.91$. The learned gate is evaluated on a distinct suite (Table 21), making its regret conceptually comparable but not strictly paired.

F CONTROLS FOR THE ROUTER AND THE EXPERT

This appendix collects the controls behind §4.3. It opens with the router's boundary and how the router behaves when the serving prompt drifts, then turns to matched swaps: each of §F.1–§F.4 replaces one component of the pipeline, holding everything else fixed, and re-measures.

The boundary, and the near-misses. The multiclass router, trained on the repaired records of Appendix B.3, fires its TorchHub class on 77% of TorchHub requests under TorchHub's own serving prompt and sends none to TF Hub. While alternative methods can force safe outputs, they severely degrade accuracy on similar datasets: masked gated couplings never invent names but are legal-but-wrong on foreign datasets, costing 6 points on the sibling dataset, and proxy-in-pool limits performance to the proxy's baseline. Simply failing safely is not true routing. Instead, evaluating the entire request remains superior; the best per-step gate at one setting leaves 13.2 points of regret, compared to the request-level router's 3.0 (Table 24). This separation holds even at segment granularity (Appendix G).

The router under serving-prompt drift. Modifying the input prompt across five severity levels (Table 26) reveals the router keeps own-class activation at 72 to 78% through L3. It fails, defaulting on 93% of requests, only when the hub framing is entirely removed. This confirms the router relies on the hub frame to recognize its catalog, as its own-class activation drops to 2.5% without it. However, when forcing all datasets into a single shared prompt template, where a rule reading only the frame would apply the TF Hub strategy to every request, performance trends remain robust ($+0.0$ / $+0.0$ / $+0.5$ on HuggingFace / TorchHub / GSM8K). This closely tracks the baseline results from Table 5 ($+0.0$ / $+0.0$ / $+0.0$), demonstrating the router does not naively rely on the frame alone.

The contract under one prompt. Because a ubiquitous hub frame degrades general answers (Table 28), Table 25 evaluates requests bare (without framing) with every inventory armed. Under bare prompts, neither head reliably fires the TF Hub or HuggingFace class (0.1 to 14.4% activation), showing the inventory signal requires the serving frame to meet condition (A). On bare prompts, the deployed head defaults on 96.5 to 97.8% of TF Hub and HuggingFace requests and loses at most 0.4 points; the refit head defaults on 71.5 to 96.2% of bare requests and loses at most 0.5. TorchHub remains anomalous, as the deployed head activates on 34.9% of bare requests for a 0.5-point gain. Consequently, the system is deployed framed: the router evaluates requests within the application's serving frame, and a default verdict returns the host's answer to the request as it arrived. Stripping the frame on a default verdict is an optional app-side step; Table 28 measures it.

The serial deployment on closed hosts. Table 27 details our tests on closed-API models, supporting the T3 composition of §4.3. We evaluated a 1,764-request mixed stream across three catalogs, each

head	bed (n)	own-class, own prompt	own-class, bare	default, bare	foreign class, bare	composed, own prompt	composed, bare
deployed	TF Hub (688), host 49.1	77.2	1.5	96.5	2.0	64.1 (+15.0)	48.7 (−0.4)
refit	TF Hub (688), host 49.1	81.4	14.4	84.4	1.2	65.6 (+16.4)	51.9 (+2.8)
deployed	HuggingFace (911), host 3.6	85.5	0.1	97.8	2.1	16.5 (+12.9)	3.5 (−0.1)
refit	HuggingFace (911), host 3.6	80.1	0.5	96.2	3.3	15.8 (+12.2)	3.9 (+0.3)
deployed	TorchHub (186), host 7.7	77.4	34.9	64.5	0.5	11.5 (+3.8)	8.2 (+0.5)
refit	TorchHub (186), host 7.7	88.2	21.5	71.5	7.0	13.7 (+6.0)	7.1 (−0.5)

Table 25: The two-condition contract under one prompt. We report verdict shares (%) and composed accuracy on each hub's native dataset with all inventories armed. The deployed and refit heads evaluate questions under both the hub's serving prompt and a bare prompt. For composition, TF Hub uses the unmasked coupling (always-on accuracy: 69.8), while HuggingFace and TorchHub use the merged expert under a flat mask (17.9 and 13.1). Verdict shares are calculated over all evaluation questions (688, 911, and 186, respectively). Accuracy is evaluated over the questions whose gold has a parsable identifier (688, 893, and 183).

serving prompt	own (TF)	sibling (Torch)	HF	BFCL	default
L0 trainer-exact	**77.8**	0.6	2.0	0.1	19.5
L1 light paraphrase	72.2	0.6	1.6	0.0	25.6
L2 restructured	71.8	0.0	0.9	2.8	24.6
L3 minimal, hub named	76.6	0.0	0.0	6.0	17.4
L4 generic, no hub frame	2.5	4.2	0.0	0.6	**92.7**

Table 26: Router verdict shares (%) as the serving prompt drifts, TF Hub frame, $n = 688$. Rows sum to 100. Without the hub frame, the TF Hub class fires on 2.5%.

evaluated under its native serving prompt without abstention requests. The system applies the restrictive mask exclusively when the local router identifies a TF Hub request; misapplying this mask to a foreign query automatically yields a score of zero. Despite this strict penalty, all five tested models performed within 0.6 points of a perfect oracle router. Only gpt-4.1-mini gained nothing over free-form generation. This occurs because its free-form accuracy on TF Hub already nearly equals its masked accuracy in this run (+3.1 in the re-measured snapshot of Table 19).

closed host	free-form	oracle router	local router
gpt-4.1	20.6	28.7	28.1
gpt-4o-mini	16.9	25.4	24.9
gpt-4o	23.3	27.9	27.6
o4-mini (low)	24.1	28.4	27.9
gpt-4.1-mini	27.1	27.7	27.1

Table 27: The serial deployment: identifier-match accuracy (%) over the 1,764-request mixed stream of the three hubs. Free-form: no mask. Oracle: the TF Hub structured-output mask on TF Hub requests only. Local router: the mask on iff the multiclass verdict is TF Hub.

The default class off its training distribution. The router's default class was trained exclusively on math questions, generalizing poorly to other general requests. Under a hub prompt, it incorrectly routes most general tasks to expert models (Table 29). Retraining the router with diverse general examples successfully reduced these errors to $\leq 10\%$, though it still misrouted 17–20% of a completely unseen chat dataset. Furthermore, this broader retraining introduces a trade-off. While it fixes the general tasks it was fitted on under a hub frame, it misfires on 31.1% of bare HumanEval requests (a 20.1-point loss), where the deployed head never fires. Consequently, the deployed head in all other tables remains the original math-fitted version. Two tables detail these dynamics. Under a hub serving prompt, the prompt asks the host itself for an API call, which ruins general answers before the router even acts (Table 28). Evaluated bare (without the frame), the deployed head never fires on MMLU (0.0% activation) and HumanEval (0.0%) but misfires on 3.5–19.0% of instruction, summary, and chat requests (Table 29). The refit head suppresses these false activations to $\leq 2.0\%$, except on HumanEval. Table 28 translates these activations into realized accuracy. Without a hub frame, the deployed router's accuracy drop is at most its false activation rate, and equals it on the open-ended datasets. Under a hub frame, the refit router, if the app strips the frame on a default verdict, loses at most 1.8 points on GSM8K, MMLU, HumanEval, and CNN/DailyMail, 9.5 on Alpaca, and up to 20.0 on chat (Dolly), an improvement over an always-on inventory strategy, which would lose everything. Finally, these tests evaluated the host using a 320-token budget. This explains the GSM8K score of

83.5, compared to 67.0 in Table 24, where a stricter 256-token budget prematurely truncated a third of the answers.

bed (metric)	host, bare	deployed head composed, bare	refit head composed, bare	refit head composed under the TF / HF / Torch frame, default answers bare	host under the TF / HF / Torch frame
GSM8K (exact match)	83.5	83.5 (exact)	83.5 (exact)	83.5 / 83.5 / 83.0	1.0 / 0.0 / 3.0
MMLU (letter match)	52.5	52.5 (exact)	52.5 (exact)	52.5 / 52.5 / 52.5	1.0 / 2.5 / 6.5
HumanEval (pass@1, executed)	62.8	62.8 (exact)	42.7 (−20.1) [−25.6, −14.0]	62.8 / 62.8 / 61.0	0.0 / 0.6 / 0.0
Alpaca (answered, not an API call)	100.0	90.5 (−9.5) [−13.5, −5.5]	100.0 (exact)	91.5 / 91.0 / 90.5	4.0 / 7.5 / 17.5
CNN/DailyMail (answered)	100.0	96.5 (−3.5) [−6.5, −1.0]	99.0 (−1.0) [−2.5, +0.0]	100.0 / 100.0 / 100.0	0.0 / 0.0 / 0.0
Dolly, unseen by both heads (answered)	100.0	81.0 (−19.0) [−24.0, −13.5]	98.0 (−2.0) [−4.0, −0.5]	83.5 / 80.0 / 82.5	1.5 / 8.0 / 14.0

Table 28: Realized accuracy (GSM8K, MMLU, HumanEval) and response preservation (Alpaca, CNN/DailyMail, Dolly: answered, not scored for correctness) on the general beds of Table 29 (Qwen2.5-7B host, greedy, same seed-0 questions); composed cells apply each head's verdicts, brackets are paired bootstrap deltas against the bare host; the framed column applies the refit head's verdicts under each frame with a default verdict answering bare (the optional stripping step), where an inventory strategy applied to every request would score 0.

	deployed head (default fitted on GSM8K)				refitted head (broad default)			
bed (n)	TF Hub prompt	HF prompt	Torch prompt	bare	TF Hub prompt	HF prompt	Torch prompt	bare
GSM8K (200), the fitted bed	0.5	0.5	1.0	0.0	0.5	0.0	0.5	0.0
MMLU (200)	16.0	14.0	19.0	0.0	0.0	0.0	0.5	0.0
HumanEval (164)	65.9	91.5	90.9	0.0	1.8	0.6	4.9	31.1
Alpaca instructions (200)	74.5	85.5	79.5	9.5	8.5	9.0	9.5	0.0
CNN/DailyMail summaries (200)	43.5	99.5	65.5	3.5	0.0	0.0	0.0	1.0
Dolly (200), unseen by both heads	82.0	92.5	74.5	19.0	16.5	20.0	17.5	2.0

Table 29: The frozen router on general beds: share of requests (%) given any inventory verdict (TF Hub, HuggingFace, TorchHub, or BFCL) instead of default, for the deployed head and for the same head refitted with broad general negatives in its default class. A default verdict returns the host unchanged, so this share bounds the composed loss on the bed. Dolly is a chat bed neither head saw.

Pool order A and B. The order of operations for constructing the candidate pool, whether selecting the top-k proposals before masking for legality (Order A) or masking before selection (Order B), yields indistinguishable results. Evaluated with the low-recall inventory-only expert at k=10 and k=50, the reset strategy produces nearly identical deltas and confidence intervals (e.g., -1.2 $[-5.1, +2.8]$ versus -0.9 $[-4.8, +2.9]$ at k=10). Furthermore, the aggregation strategy results match exactly to the decimal ($+8.7$ $[+5.8, +11.5]$ across all tested cells). At these pool sizes, the final legal intersection provided to the expert almost never diverges between the two orders.

F.1 The representation control

To isolate the effect of tuning, we trained the exact same router on the *untuned* 0.5B model's prefill states. We also established two off-the-shelf baselines trained on the same prompts: a logistic head and a 10-nearest-neighbor (10-NN) vote, both over all-MiniLM-L6-v2 sentence embeddings. Basic topicality is easily achievable; the catalog margin comes from tuning, as the untuned twin loses 14 to 31 points of catalog activation (Table 30). The 10-NN router identifies TF Hub requests nearly as effectively as the tuned head (71.1% versus 77.2% own-class activation; composed 63.0 versus 66.0 on the first 200 questions) and routes perfectly on GSM8K. This demonstrates that generic embeddings suffice for catalog-level routing on TF Hub (Table 30). The tuned head's primary advantage lies in HuggingFace recall, where it achieves 85.5% compared to the baselines' 57.6% and 68.9%. On general traffic, the tuned head's margin disappears. When trained with the refit head's 4,164 broad negative examples, the 10-NN router falsely triggers on only 0 to 14.0% of general requests under a TF Hub frame, compared to 0 to 16.5% for the refit head. Simultaneously, the 10-NN router maintains strong TF Hub routing (69.5%), though its HuggingFace recall drops to 53.6%. Finally, the 10-NN router recognizes bare requests, identifying 36.9% of TF Hub and 45.3% of HuggingFace queries. In contrast, the expert heads activate on at most 14.4% of these bare requests because the expert's prefill signal is heavily bound to the serving frame (Table 25).

router input	TF routing	HF routing	BFCL	GSM→default	composed on TF
prefill states of the tuned merged expert (deployed)	77.2	85.5	99.8	100	**66.0**
prefill states of the untuned base (control)	62.8	54.7	100	100	61.0
MiniLM sentence embeddings, logistic head (same prompts, same rule)	49.9	68.9	—	100	57.0
MiniLM sentence embeddings, 10-NN vote	71.1	57.6	—	100	63.0

Table 30: The representation control: correct-routing rates (%) and the composed accuracy on the expert's catalog (the k=200 coupling on the first 200 questions, host 48.5). The untuned twin keeps the topicality classes and loses 14 to 31 points of catalog activation, 5 composed points.

F.2 THE EXPERT'S OWN BASELINE

We evaluate our expert against standard supervised retrievers trained on the same log records. These baselines represent the standard stack for question-to-identifier mapping, including sparse (Robertson & Zaragoza, 2009), dense (Reimers & Gurevych, 2019; Karpukhin et al., 2020), and benchmark-specific retrievers (Patil et al., 2024). Our model outperforms the best retriever by a wide margin (68.8 versus 56.8). Crucially, this advantage stems entirely from the tuned language model's internal ranking of the catalog. It does not come from decode-time composition. The coupling mechanism adds only one additional point (69.8, Table 4).

question→identifier method (TF, n=688)	top-1	top-5
BM25 over inventory cards (zero-shot)	19.6	56.7
embedding → nearest card (zero-shot)	52.3	70.1
embedding kNN over train questions (supervised)	56.8	82.8
per-identifier centroid (supervised)	32.7	78.9
host free-form output snapped to the nearest legal identifier	49.9	—
masked expert (host's anchor) / bare expert (the 0.5B alone)	58.9 / **68.8**	—

Table 31: The expert's baseline. We report top-1/top-5 accuracy (%) over the 688 evaluation questions of the TF Hub dataset. Every row is scored on the exact same questions, with retrievers trained on the identical log records used for the expert. While the best supervised retriever approaches the masked expert's performance, it falls 12 points below the bare expert. Additionally, we evaluate a post-hoc string-snapping baseline, which maps the host's free-form identifier to the nearest legal string. Snapping adds only 0.8 points (from 49.1 to 49.9). This falls well below the 53.5 accuracy achieved by this run's flat mask. Although both methods guarantee a legal identifier, the mask constrains the decoding process at every slot, whereas snapping merely repairs a fully extracted string.

Cards and the expert do not stack. We tested augmenting the prompts of both models with retrieved context cards (Table 32). This combination fails to improve upon the better of the two methods used in isolation. On TF Hub, introducing cards degrades the expert-in-mask accuracy by 10 points (from 69.0 to 59.0). This regression likely occurs because the expert was trained exclusively on prompts without card context, making the augmented prompt out-of-distribution. Across the remaining datasets, combining the expert with retrieved cards yields zero improvement or actively harms results. Furthermore, on HuggingFace, the mask itself struggles to process card context. We hypothesize this occurs because the additional text causes the model to prematurely commit to an incorrect prefix during the decoding process.

Few-shot prompting. The other way to give the host the catalog is to put examples in its prompt, so we measure it: five BM25-selected solved *training* questions, against a zero-shot control run in the same process and build, so the difference is not a cross-run artifact. It lifts free-form generation on TF Hub and HuggingFace without substituting for the expert (Table 33).

bed (n)	cards + host	cards + anchored mask	cards + expert in mask	cards + in-pool sum	expert + mask, no cards	host, no cards
TF Hub (300)	56.3	59.7 (+3.3) $[+1.0, +5.7]$	59.0 (+2.7) $[-2.3, +7.7]$	62.3 (+6.0) $[+3.0, +9.3]$	**69.0**	48.7
HuggingFace (300)	16.3	12.0 (-4.3) $[-7.7, -1.3]$	14.3 (-2.0) $[-6.0, +2.0]$	13.3 (-3.0) $[-6.3, +0.0]$	17.0	3.3
TorchHub (183)	18.0	18.6 (+0.5) $[-1.1, +2.2]$	12.0 (-6.0) $[-10.4, -1.6]$	19.1 (+1.1) $[-2.7, +4.9]$	13.1	7.7

Table 32: Performance of the expert and retrieved cards evaluated on matched questions. We evaluate the 7B host and merged expert under the factorial protocol on the first 300 evaluation questions of TF Hub and HuggingFace, and all 183 questions of TorchHub. The four card-augmented columns share a single prompt containing the dataset's top-5 dense-retrieved cards. Parentheses denote paired deltas against the baseline cards + host cell. The final two columns display the published card-free runs on these exact same questions for comparison. Finally, we clarify a minor discrepancy regarding Table 3. The first two card columns in Table 3 derive from the serving harness (a second decoder), while its third column matches this table's cards + expert in mask configuration. On TorchHub, the only dataset scored in full across both tables, these two harnesses agree exactly. The TF Hub and HuggingFace rows in this table represent 300-question subsets rather than the full evaluations.

	TF Hub (688)	HuggingFace (893)	TorchHub (183)
free-form, zero-shot (in-run control)	49.1	3.7	7.7
free-form, 5-shot BM25	62.8	16.8	8.2
5-shot under the hub mask	62.8	6.9	9.8
in-catalog share of the free-form answers	95.8%	74.0%	86.3%
bare expert (the 0.5B alone)	**68.8**	—	—

Table 33: Few-shot prompting as a baseline (%, same harness and build as the zero-shot control). It lifts free-form generation by $\sim$13 points on TF Hub and HuggingFace but does not substitute for the expert: 4–26% of its free-form answers name identifiers that are not in the catalog.

F.3 The scorer's token family and architecture

In this section, we evaluate whether the expert's fundamental architecture can be altered. At the identifier slot, the coupling mechanism requires neither the host's token family nor a generative language model. It only requires a scorer that has learned the catalog. Because candidates are evaluated as whole strings, any scorer can rank them using its own native tokenizer within the host's top-k legal pool. Scoring whole candidates makes this cross-tokenizer integration mathematically well-defined.

What the swap settles. Swapping the expert model yields three key insights (Table 34). First, the expert is not required to generate text; a simple cross-encoder actually achieves the highest performance. Second, the host and expert do not require matching tokenizers, as models with foreign tokenizers perform equally well. Third, catalog training accounts for the vast majority of the performance gain. The best and worst rows in the table share the same base model (BERT), trained on the catalog as a cross-encoder and used off the shelf as an embedder. This confirms the expert's contribution relies primarily on catalog-specific learning. Finally, we report conservative baseline numbers here; settings tuned individually per scorer yield even higher scores, though these are best-of-seven maxima.

Scope of the claim, and a length-bias check. This appendix does not claim to solve general token-level cross-tokenizer alignment. Instead, it establishes that *whole-candidate scoring allows a foreign-tokenizer scorer to operate successfully at the identifier slot*. This operates at a different granularity than the token-by-token merge described in §3. Moving from token-level to whole-candidate granularity incurs a performance tax, dropping accuracy from 67.0 (expert-decides at k=10 in Table 16) to 62.1 here. We also investigated whether these results suffer from length bias. Because a candidate's score is a sum of log-probabilities, the model might simply favor shorter strings. To test this, we applied a Pointwise Mutual Information (PMI) variant, subtracting a question-free score to cancel the length prior. The point estimates maintained their exact ranking, with the foreign SmolLM

scorer	class	r@10	best agg	Δ vs masked host	vs its own untuned base
BERT-base cross-encoder	encoder, trained	**69.2**	—	**+19.3** [+16.4, +22.5]	+39.7
T5-base encoder	encoder, trained	67.7	—	+17.9 [+15.1, +20.8]	—
ALBERT-base-v2	encoder, trained	67.3	—	+17.4 [+14.7, +20.5]	—
Falcon3 1B	Falcon family	63.5	**65.0**	+13.7 [+10.6, +16.7]	+26.0 [+22.4, +29.8]
SmolLM2 360M	GPT-2 family	62.4	63.8	+12.5 [+9.4, +15.6]	+24.4 [+21.1, +28.2]
Qwen2.5 0.5B *(in-family)*	host's family	62.1	64.8	+12.2 [+9.0, +15.4]	+23.1 [+19.2, +27.0]
MiniLM embedder	encoder, off-the-shelf	38.1	—	−11.8 [−16.0, −7.6]	—
BERT embedder	encoder, off-the-shelf	29.5	—	−20.3 [−24.0, −16.7]	—
Qwen2.5 0.5B	decoder, untuned	39.0	52.8	−10.9 [−14.8, −7.4]	—
SmolLM2 360M	decoder, untuned	37.9	38.2	−11.9 [−15.7, −8.1]	—
Falcon3 1B	decoder, untuned	37.5	38.2	−12.4 [−16.1, −8.7]	—

Table 34: The scorer swap on TF Hub ($n = 688$, catalog 80, masked host 49.9). r@10: accuracy when the scorer alone picks among the host's top-10 legal candidates under the pre-registered reset rule, the same pool for every row (gold-in-pool@5/10/50 is 75.1/81.2/97.7). Best agg: the best of three host-plus-scorer weights ($\beta \in \{0.5, 1, 2\}$) over the host's top-50, chosen on these questions. The last column compares each scorer with its own off-the-shelf or untuned twin. An untuned TinyLlama, left out as possibly contaminated (untested), reaches 55.4 at r@10, $+5.5$ $[+2.2, +8.7]$ over the masked host.

($+10.2$) still outperforming the in-family Qwen ($+8.6$), though the intervals overlap. However, PMI performs worse in absolute terms. By removing the model's marginal prior over catalog entries, PMI collapses the host baseline from 49.9 to 28.8. Because the host's candidate pool is also re-ranked under PMI, this similarly collapses the pool ceiling from 81.2 to 57.4.

F.4 The second model family

To ensure our findings generalize, we rebuilt the coupling and router end-to-end on the Falcon3 family, utilizing a 10B host and a 1B expert (Table 35). Using the same training recipe (on the unrepaired records of Appendix B.3), the core coupling mechanics replicate. The TF-only expert gains $+24.0$ $[+20.3, +27.9]$ always-on accuracy on TF Hub. Interestingly, in-pool aggregation outperforms reset for the TF-only Falcon expert, reversing the Qwen ordering observed in Table 4. The router keeps most of the always-on gain at one setting ($+16.6$ of $+20.6$ in Table 35). It defaults exactly on math questions (0 of 200 TF-class verdicts on GSM8K). Under TorchHub's own prompt, the TF class incorrectly fires on 6.5% of Falcon's requests and on none of Qwen's. The Falcon3 router was trained on the unrepaired TorchHub records of Appendix B.3, the Qwen router on the repaired ones.

G Mixed-intent requests: a first probe of segment-level routing

The router of §3 issues one verdict per request. A mixed-intent request, whose parts want different treatment, breaks that granularity.

Beds. Synthetic mixed requests built from the evaluation questions: 200 two-part prompts (a TensorFlow Hub question and a GSM8K question), 100 three-part (adding HuggingFace), and 100 four-part with one part per class (TensorFlow Hub, HuggingFace, TorchHub, GSM8K), joined by a connective template. Each part is scored by its own metric (gold identifier for the catalogs, final answer for arithmetic), so the unit of accuracy is the *part*.

	Falcon3 (10B + 1B)	Qwen2.5 (7B + 0.5B)
always-on, own catalog	+20.6 [+17.2, +24.6]	+18.5 [+15.3, +21.7]
router-composed, own catalog	+16.6 [+13.5, +20.1]	+15.3 [+12.4, +18.2]
own-catalog activation	75.1%	77.2%
TF-class activation on the sibling bed, own prompt (Torch)	6.5%	0.0%
TF-class activation on the general bed (GSM8K)	0 / 200	0 / 200
wrong-class misfires under the TF prompt (HF / Torch)	2.7% / 47.3%	3.2% / 1.1%

Table 35: The second model family: the coupled system and its router rebuilt on Falcon3 with the same recipe (Falcon3 on the unrepaired TorchHub records, Qwen2.5 on the repaired ones, Appendix B.3). Deltas vs each family's own host on TF Hub, all 688 questions: Falcon3 couples host and expert ($\beta = 1$) under the anchored mask, Qwen2.5 runs the merged expert alone under the flat mask. The last row is the torch sibling residual.

Segmenting router. A linear head over the frozen merged 0.5B expert's last-layer prefill states classifies each *sentence* of the request in full context (TensorFlow Hub / HuggingFace / TorchHub / *default*); contiguous same-class sentences form blocks, and each block is routed to its handler: the expert-and-mask machinery for its own catalog, the untouched host for default. Training uses 4,000 mixtures of arity 1–4 built only from train-side questions, with varied connective templates and the evaluation template held out; no eval question or template is seen in training. Table 36 gives the per-part accuracies.

bed (parts)	single verdict	self-segmented	oracle split	part routing	boundary recall
2-part TF+GSM (400)	24.2	**63.2**	67.0	95.5	91.0
3-part TF+GSM+HF (300)	17.0	**43.8**	52.5	78.3	77.0
4-part all four (400)	12.2	**33.3**	41.6	61.8	71.3

Table 36: Per-part accuracy (%) on the mixed-intent beds. *Single verdict* is the request-level router; *oracle split* composes each part's best handler at the true boundaries. *Part routing* is the fraction of parts sent to the correct handler; *boundary recall* the fraction of true part starts recovered within ± 15 characters. The self-segmented and oracle cells of the 3- and 4-part beds are scored on 299 and 399 parts (one HuggingFace question has no parsable identifier; on the same parts the single verdict scores 17.1 and 12.3).

Findings. First, the single verdict stands down entirely on mixed traffic: 100% *default* verdicts on every bed, so it gains nothing, but it also never pays; forcing the expert over the whole request instead collapses with arity (10.5% at two parts, 1.8% at four). The request-level failure is total but safe. Second, the segmenting router recovers 72–91% of the oracle-vs-single-verdict gap, decreasing with arity; the loss includes class confusion among sibling catalogs at part level (TorchHub parts route to their siblings; Appendix B.3), and the share due to boundary errors is not isolated. Third, the token-level control fails the same job: a segmenter reading the per-position KL between each single-inventory expert and the untuned base, given a per-bed charitably swept threshold, reaches only 50.0/54.7/42.8% part routing (vs. 95.5/78.3/61.8% for the head) and assigns the TensorFlow Hub and TorchHub classes with 0–1% recall; on the two-part bed its best operating point refuses to segment at all (0.8% boundary recall). The separation of §4.3 reproduces at segment granularity: the routing signal is far weaker in token-level statistics than in the expert's representation, within a single request as across requests.

Scope. These are synthetic beds on one host family, and the probe splits at sentence boundaries only; sub-sentence segmentation, natural mixed traffic, and the full mechanism are follow-up work.