FORGE: Form-Optimal Routing of Grounded Evidence for Frozen LLM Agents

Xi Xiao[1], **Yunbei Zhang**[2], **Chen Liu**[3], **Lin Zhao**[4], **Jialin Chen**[3], **Tianchen Zhao**[5], **Xiang Xu**[5], **Youngeun Kim**[6], **Tianyang Wang**[1], **Min Xu**[7]

[1]University of Alabama at Birmingham [2]Tulane University
[3]Yale University [4]Northeastern University [5]Amazon AGI
[6]Korea University [7]Carnegie Mellon University

✉ xxiao@uab.edu

Website Code

ABSTRACT

In agentic AI systems, frozen foundation models are increasingly deployed as closed-weight API endpoints, making downstream adaptation possible only through the inputs and inference procedures surrounding the model. As a result, for each input query, two coupled decisions largely determine both answer quality and token cost: *what evidence to provide* and *how much reasoning budget to allocate*. Fixed defaults along these axes are often suboptimal, misallocating support form or reasoning depth on roughly 80% of queries in our analysis. To address this challenge, we propose FORGE, a unified framework for adapting frozen models through per-query routing over a joint action space that spans both support form and thinking depth. Under an entropy-regularized, cost-aware utility objective, we derive a closed-form Boltzmann routing target and instantiate the policy as a lightweight 269K-parameter factorized router. The routing policy is trained around the frozen host, without any weight access, through a three-stage pipeline: offline arm enumeration, supervised Kullback-Leibler (KL) distillation from the Boltzmann target, and Group Relative Policy Optimization (GRPO) refinement with host feedback. Across 5 knowledge-intensive benchmarks and 8 frozen backbones ranging from 7B to 671B parameters, FORGE improves accuracy at 42–45% lower token cost on both main hosts, transfers zero-shot across hosts at lower token cost, and composes with intrinsic thinking budgets where available.

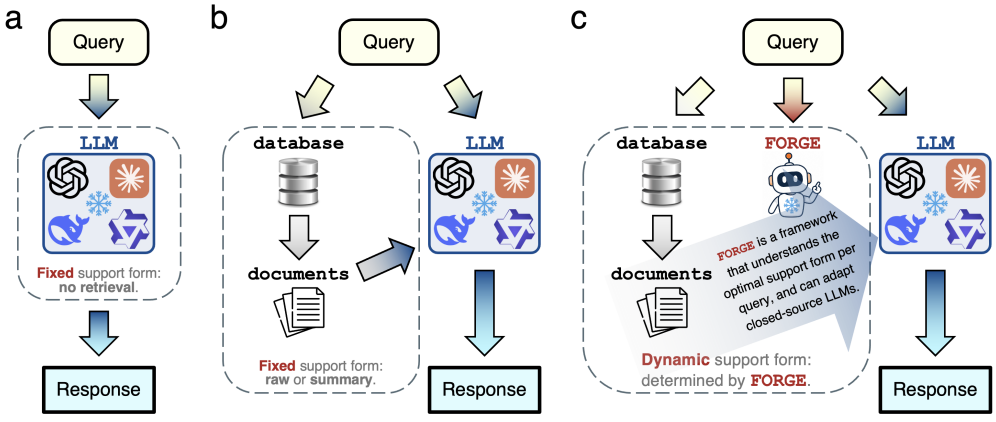

Figure 1: Fixed support forms in frozen-host workflows (a,b) and per-query support-form selection with FORGE (c).

1 INTRODUCTION

Agentic AI has seen remarkable progress in recent years, with many of the most capable systems built on top of large foundation models (Achiam et al., 2023; Comanici et al., 2025; Sapkota et al., 2025; Ferrag et al., 2025). Aligning these general-purpose models with task-specific requirements usually relies on adaptation via fine-tuning, using methods such as low-rank adaptation (LoRA) (Hu et al., 2022) and residual adapters (Rebuffi et al., 2017). However, foundation models are increasingly deployed as closed API endpoints, with private weights and no access to internal states, making host-side fine-tuning infeasible (Sun et al., 2022; Cheng et al., 2024a; Gao et al., 2023; Asawa et al., 2025). This raises a central question: how can we effectively adapt frozen, closed-weight foundation models to downstream tasks?

For a fixed frozen host, effective adaptation hinges on two axes: *what evidence to provide* (Jeong et al., 2024; Asai et al., 2024; Zhu et al., 2026; Lewis et al., 2020b; Karpukhin et al., 2020), and *how much reasoning budget to allocate* (OpenAI et al., 2024; Guo et al., 2025; Anthropic, 2025; Wang et al., 2025; Wei et al., 2022; Snell et al., 2024; Press et al., 2023; Trivedi et al., 2023; Yue et al., 2025; Muennighoff et al., 2025). In practice, however, agentic AI workflows commonly rely on fixed forms of support applied uniformly across queries, such as direct inference or retrieval-augmented generation with a predefined evidence format (Figure 1). This uniform treatment can be suboptimal because the most effective support form and reasoning budget vary substantially across queries. A controlled probe of a frozen `Qwen3-8B` on `HotpotQA` (N=2,500) makes this gap concrete (Figure 2). On the evidence axis, 7.3% of queries are answered *more* accurately without retrieval than with full passages, 79.4% achieve identical F1 under a query-aware compressed summary as under the full top-k, and only the remaining 20.6% show any F1 difference between the two. On the reasoning axis with `Qwen3-8B`, 12% of queries are *harmed* by long reasoning, 60% reach identical F1 under NOTHINK as under THINK-HIGH, and only 28% genuinely benefit. These results suggest that both axes call for adaptive, per-query decisions. Beyond this, the two decisions are inherently coupled: better evidence reduces the reasoning needed to reach an answer, and deeper reasoning reduces the evidence required to support it. Addressing either alone leaves this coupling unexploited.

This observation raises three open questions. ① **How should the two axes be jointly adapted at the query level?** Adaptive-retrieval and adaptive-reasoning methods each address one axis while fixing the other, leaving no formulation that adapts both together per query. ② **How can such a joint policy be trained around a frozen, weight-inaccessible host?** Since the host cannot be finetuned, adaptation must rely on a separate router that learns from external signals such as offline labels and online host feedback. However, existing methods treat these two signals as separate training paradigms, rather than as complementary signals for a common cost-aware objective. ③ **What target should such a policy be trained toward?** Existing routers define training targets heuristically, without a closed-form connection to the underlying cost-aware decision problem.

To answer ①, we formalize joint adaptation as per-query routing over a unified action space. The action $a = (\mu, \theta)$ pairs a *support form* $\mu \in \{\text{Direct}, \text{Summary}, \text{Raw}\}$, corresponding to three

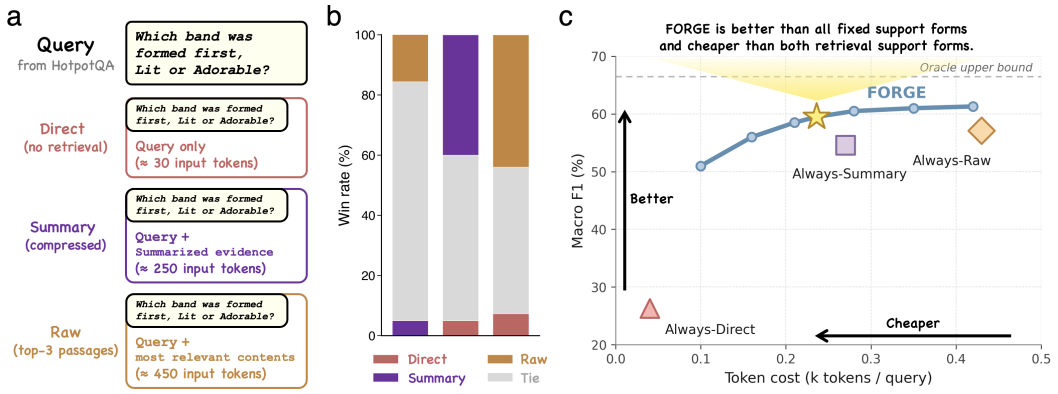

Figure 2: **Frozen LLM agents are picky readers. a.** The three common support forms. **b.** No support form is universally optimal; the best choice depends on the input query. **c.** Our proposed FORGE establishes a new Pareto frontier over fixed support forms.

evidence-channel rates between the corpus and the host (no retrieval, query-aware compressed retrieval, and full top-k passages), with a *thinking depth* θ (two settings on non-thinking hosts and four on hosts that expose test-time compute control). For each query, the router selects a single joint action a that maximizes the Pareto utility $U(x, a) = F_1(x, a) - \lambda_{\text{in}}\tilde{c}_{\text{in}}(a) - \lambda_{\text{out}}\tilde{c}_{\text{out}}(x, a)$, trading answer F1 against normalized input and output token cost. Crucially, the two axes are no longer optimized in isolation: a single utility evaluates the joint choice and a single policy outputs it per query. Within this formulation, FORGE instantiates the joint policy as a 269K-parameter factorized router that selects the action in a single forward pass before the frozen host answers. Importantly, standard RAG, Adaptive-RAG, and tiered-memory routing arise as degenerate special cases of the proposed FORGE by fixing one or more routing dimensions.

To answer ②, we train the router through a three-stage pipeline that runs entirely around the frozen host without changing its weights. Stage 0 enumerates a 3-arm warm-start subset $\mathcal{A}_{\text{warm}} = \mathcal{M} \times \{\text{NoThink}\}$ on $1{,}500$ training queries, recording per-arm F1 and token cost ($\approx 4{,}500$ host calls, under one hour). Stage 1 fits the warm-start router π_{θ_0} to a Boltzmann target restricted to $\mathcal{A}_{\text{warm}}$ via KL distillation (CPU-only, two minutes). Stage 2 lifts the policy to the full alphabet $\mathcal{A}$ via GRPO (Shao et al., 2024) with sampled host rewards and a KL anchor to π_{θ_0} (8–12 hours on one 8-GPU node). The same pipeline produces the 6-arm router for non-thinking hosts, the 12-arm joint-policy router that composes with the reasoning-budget axis on capable hosts, and a single router that transfers zero-shot across hosts. Router computation adds 0.5–5.5 ms per query; on a fresh query, the full variant also issues four parallel host probes for its features, whereas FORGE-Lite needs only the final answer call (§4.2, App. B.9).

To answer ③, we ground the router in a unified Pareto reward and a closed-form Boltzmann target. The Pareto utility above is the system's single source of truth: every training stage, every baseline, and every Pareto operating point sweeps along it. Adding an entropy regularizer to the cost-aware utility maximization, as in maximum-entropy reinforcement learning (Ziebart et al., 2008; Haarnoja et al., 2018), fixes a unique closed form on the discrete action space, $\pi^*(a \mid x) \propto \exp(U(x, a)/\tau)$. The same softmax form is also the choice rule of discrete-choice random-utility models (MCFADDEN, 1973), suggesting that the Boltzmann target is a natural cost-aware decision rule rather than an ad hoc soft label. Stage 1 distills this target on the warm-start actions, and Stage 2 refines the policy against the same utility with sampled host rewards; App. D characterizes the idealized fixed point of the KL-anchored Stage-2 objective. Proposition 3 (App. D.3) further decomposes the warm-start mismatch into a support-form marginal gap plus a closed-form $\log|\Theta| - \mathcal{H}$ cost from the uniform thinking head. These results characterize the target and its initialization rather than guarantee convergence of the implemented optimizer.

Across 5 knowledge-intensive benchmarks and 8 frozen LLM backbones from 7B to 671B parameters across three deployment regimes (non-thinking local, thinking-capable, frontier non-thinking), FORGE Pareto-dominates Always-Raw on every cell tested on `Qwen3-8B` and `Mistral-7B` ($+2.4$ and $+2.9$ macro F1 at 45% and 42% lower cached token cost; §4.2). A single router trained on `Qwen3-8B` transfers zero-shot to three frontier backends, matching or exceeding Always-Raw F1 on 11 of 15 task / host cells at 30%+ lower cached cost (§4.6). On thinking-capable hosts the joint 12-arm policy exceeds Always-Raw with maximum thinking budget by $+1.4$ to $+1.9$ macro F1 at roughly half the total token cost (§4.5). We believe that this work contributes to the tractable, theoretically grounded, and broadly transferable axis of frozen-agent adaptation.

2 PRELIMINARIES AND RELATED WORKS

2.1 RELATED WORKS

Model routing selects the host (Chen et al., 2024; Ong et al., 2025; Jiang et al., 2025b; Cheng et al., 2024b; Su et al., 2024; Jiang et al., 2023; Hu et al., 2024; Ding et al., 2024); *adaptive retrieval* selects whether and what to retrieve (Jeong et al., 2024; Asai et al., 2024; Zhu et al., 2026); *thinking-budget control* selects reasoning effort (Wang et al., 2025; Alomrani et al., 2025; Welleck et al., 2024; Yao et al., 2023). Building on these choices, including partial support-form selection, FORGE jointly selects Direct, Summary, or Raw evidence and a host-exposed thinking setting under one measured quality–cost utility (extended discussion in App. A).

2.2 Routing Objective and Boltzmann Solution

Let A be a frozen pre-trained LLM with fixed, inaccessible parameters, x a query, and $\mathcal{D}$ a document corpus. The router selects $a = (\mu, \theta) \in \mathcal{A} = \mathcal{M} \times \Theta$, with support $\mu \in \{\text{Direct}, \text{Summary}, \text{Raw}\}$ and thinking configuration θ (§2.3). Using prompt constructor g_a, the host returns $\hat{y} = A(g_a(x, \mathcal{D}))$, with F1 $m(x, a) = \text{F1}(\hat{y}, y)$ and input/output token costs $c_{\text{in}}(a), c_{\text{out}}(x, a)$. Dataset-level maxima normalize these costs to $\tilde{c}_{\text{in}}, \tilde{c}_{\text{out}}$, yielding the per-query *Pareto utility*:

$$U(x, a) \;=\; m(x, a) \;-\; \lambda_{\text{in}}\tilde{c}_{\text{in}}(a) \;-\; \lambda_{\text{out}}\tilde{c}_{\text{out}}(x, a), \quad \lambda_{\text{in}}, \lambda_{\text{out}} \geq 0, \tag{1}$$

The weights set the quality–cost operating point; $\lambda_{\text{out}}{=}0$ penalizes input cost only. Labeled queries provide $m(x, a)$ for executed actions during training and evaluation. At inference, routing uses pre-answer features without observing the selected response's F1.

For a fixed query x, finite action set $\mathcal{A}$, finite utilities $U(x, a)$, and temperature $\tau > 0$, the entropy-regularized objective $\max_{\pi(\cdot|x)\in\Delta(\mathcal{A})} \sum_a \pi(a \mid x)U(x, a) + \tau H(\pi(\cdot \mid x))$ has the unique solution

$$\pi^*(a \mid x) \;=\; \frac{\exp\big(U(x, a)/\tau\big)}{\sum_{a'} \exp\big(U(x, a')/\tau\big)}, \tag{2}$$

As $\tau \to 0$, it concentrates uniformly on utility maximizers; as $\tau \to \infty$, it becomes uniform. The standard solution gives Stage 1 soft labels on enumerated warm-start actions. Optimality holds for the stated entropy-regularized objective, without guaranteeing that the parameterized router or clipped Stage 2 update attains it. Stage 2 refines the policy with sampled host rewards over the full combination of support forms and thinking settings (§3.2).

2.3 The Adaptive-Thinking Action Alphabet

The support form μ is *Direct*, the query alone (negligible input tokens); *Summary*, BM25 retrieval with query-aware sentence-level compression ($\approx$ 250 input tokens); or *Raw*, full top-k BM25 passages ($\approx$ 450 input tokens). Thinking depth θ is *NoThink* (plain decoding), *CoT-Prompt* (the in-context "Let us think step by step" suffix), or, when the host exposes a reasoning budget, *Think-Low* or *Think-High* (construction in App. B.3). Hosts without a reasoning mode use $\Theta_{\text{nt}}{=}\{\text{NoThink}, \text{CoT-Prompt}\}$ ($K{=}6$); thinking-capable hosts use $\Theta_{\text{t}}{=}\{\text{NoThink}, \text{CoT-Prompt}, \text{Think-Low}, \text{Think-High}\}$ ($K{=}12$).

Query-dependent utility maximizers motivate adaptive routing. Proposition 1 (App. D.1) specifies when a per-query utility oracle strictly exceeds every fixed action in expectation; §4 and §4.6 examine action heterogeneity and six-arm references. The *Oracle* rows are descriptive references: their selection rule was not retained, precluding claims of utility optimality or F1 upper bounds.

3 Method

FORGE jointly controls evidence delivery and reasoning depth before the frozen host answers. The insight is their interdependence: evidence quality changes the value of extra reasoning, while the reasoning setting changes what support is worth providing. Candidate actions' F1 and input/output tokens define the utility in Eq. (1) and soft targets in Eq. (2). Figures 3 and 4 summarize training and routing.

3.1 Router and Deployment Features

The policy factors into a support-form head and a support-conditioned reasoning head:

$$\pi_\theta(a \mid x) = \pi_\theta^\mu(\mu \mid h(x))\, \pi_\theta^\theta(\theta \mid h(x), \mu). \tag{3}$$

A shared two-layer MLP maps inputs through 256 hidden units to a 256-dimensional representation (ReLU, dropout 0.1); the heads use a learned 16-dimensional support-form embedding. The six-arm Full router has approximately 269K parameters (settings in App. B.4). Full FORGE uses a frozen 768-dimensional BGE embedding and 21 structured features (789 total). Eleven features depend on one greedy host probe and three self-consistency samples, requiring *four host calls before* the routed answer on a fresh query. Caching shifts these calls outside routed-answer accounting but

does not eliminate their fresh-query cost. FORGE-Lite is retrained on 778 host-independent features: the embedding, four BM25 statistics, one query–top-passage cosine similarity, and five structural features. It requires no training or inference probes and only one final-answer host call. Both variants share the single-step action set and frozen retrieval pipeline. *Cached* counts routed-answer tokens; *Fresh Online* tokens and latency include every host call for a new query (Appendix Figure S1).

3.2 TRAINING

Stage 0: action evaluation. On each labeled warm-start query x_i, the frozen host executes $a \in \mathcal{A}_{\text{warm}} = \mathcal{M} \times \{\text{NoThink}\}$ and records $(m_i^a, c_{\text{in},i}^a, c_{\text{out},i}^a)$. These calls incur training cost without updating host weights. Gold answers provide m_i^a; §4 tests proxy rewards with gold-labeled development data for router hyperparameter selection.

Stage 1: supervised distillation. Using $U(x_i, a)$ from Eq. (1), the evidence head minimizes $D_{\text{KL}}(\pi^*_{\text{Boltz}} \| \pi^{\mu}_{\theta})$ against Eq. (2) on the three warm-start actions at $\tau_0 = 1$; the reasoning head starts uniform. A separate Stage-1-only control uses six-arm offline observations, holding the full action set fixed to isolate the contribution of Stage 2.

Stage 2: policy-guided refinement. Over the full action set, each of B queries receives G sampled actions, stratified to cover every evidence form when $G \geq |\mathcal{M}|$. Frozen-host executions yield $r_{b,g} = U(x_b, a_{b,g})$ and normalized group-relative advantages

$$\hat{A}_{b,g} = \frac{r_{b,g} - \bar{r}_b}{\sigma_{r_b} + \epsilon}, \qquad \bar{r}_b = \frac{1}{G} \sum_g r_{b,g}, \qquad (4)$$

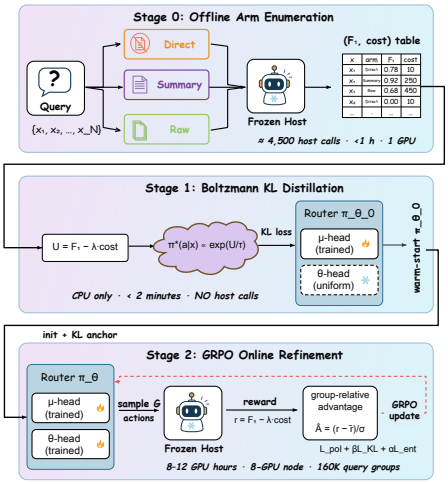

Figure 3: **Three-stage training around a frozen host.** Stage 2 uses $5{,}000 \times 32 = 160$K query groups with eight sampled actions each: 1.28M host completions.

where $\epsilon = 10^{-4}$ and zero-variance groups receive zero advantage. A PPO-clipped policy term (Schulman et al., 2017; Shao et al., 2024) uses this feedback, with a KL anchor to Stage 1 and an entropy bonus. Appendix B.4 specifies sampling and the surrogate; the same-action-set ablation (§4) isolates refinement from alphabet expansion.

4 EXPERIMENTS

4.1 SETUP AND EVALUATION PROTOCOLS

We evaluate HotpotQA (Yang et al., 2018), 2WikiMultiHopQA (Ho et al., 2020), MuSiQue (Trivedi et al., 2022), PopQA (Mallen et al., 2023), and FEVER (Thorne et al., 2018) on Qwen3-8B-Instruct and Mistral-7B-Instruct-v0.3, plus three thinking-capable and three frontier hosts. The six-arm alphabet pairs Direct/Summary/Raw with NoThink/CoT-Prompt; thinking hosts add Think-Low/High for 12 arms. BM25 retrieval is fixed; Summary deterministically extracts from the top three passages. Splits and retrieval settings are in Appendices B.6 and B.3.

We report token-level F1, exact match (EM), and cost $\bar{C} = \bar{C}_{\text{in}} + \bar{C}_{\text{out}}$ in k host tokens/query. *Cached* cost counts the routed answer with precomputed features; *Fresh Online* includes feature acquisition and the answer on a new query: four pre-route host calls for Full, none for Lite. Baselines are fixed Direct/Summary/Raw, BM25-Threshold, Adaptive-RAG (Jeong et al., 2024), TierMem (Zhu et al., 2026), s3 (Jiang et al., 2025b), and Sysformer (Sharma et al., 2025). The matched BGE+BM25-KL router uses 768 BGE dimensions, four BM25 statistics, and one query–passage cosine, with the same Stage-0 data, MLP capacity, optimizer, and development budget as FORGE, while excluding GRPO refinement and host-dependent feature probes.

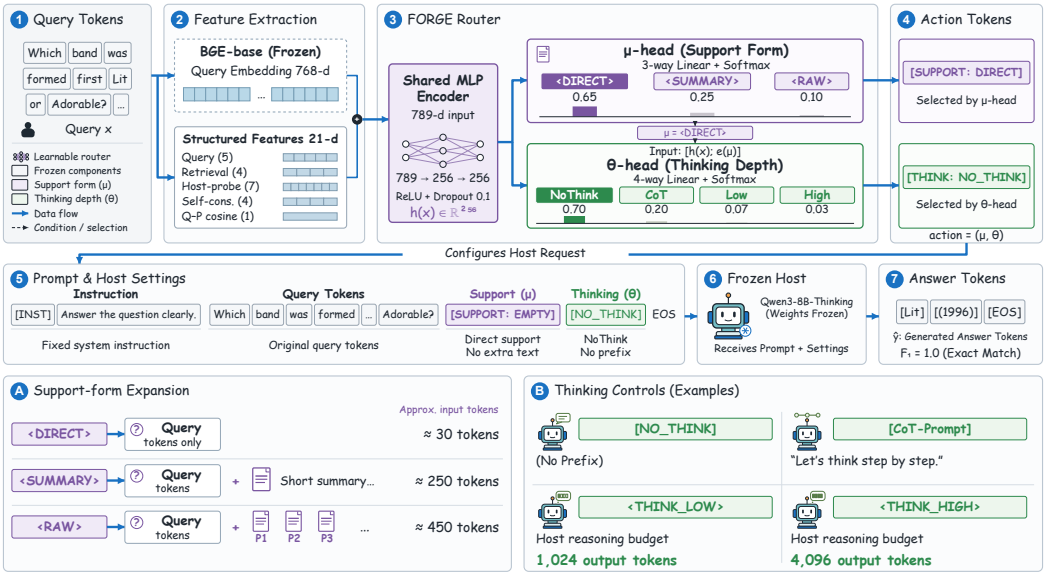

Figure 4: **Factorized router and frozen-host inference.** The $789 \to 256 \to 256$ encoder feeds support and support-conditioned thinking heads (features: Table S6). Bottom panels expand support choices and alternative settings of one host: CoT is prompted; Low/High use host-exposed budgets. Probabilities, token counts, and the answer are illustrative.

Table 1: **Canonical Cached results.** Task F1 (%) and selected-answer cost $\bar{C}$ (k/query); HQA/MSQ/Pop/FEV denote HotpotQA/MuSiQue/PopQA/FEVER. Bold/boxed values mark best practical task/macro F1 per host. Oracle is a descriptive six-arm reference; Table 2 gives online costs, including the feature calls needed before answering a new query.

Policy	Qwen3-8B							Mistral-7B						
	HQA	2Wiki	MSQ	Pop	FEV	Macro	$\bar{C}$	HQA	2Wiki	MSQ	Pop	FEV	Macro	$\bar{C}$
Always-Direct	26.2	25.7	10.2	15.0	54.4	26.3	0.040	25.2	17.7	7.9	23.0	47.0	24.2	0.041
Always-Summary	46.9	38.1	19.0	87.0	**81.8**	54.6	0.270	38.3	24.5	11.8	80.0	69.5	44.8	0.265
Always-Raw	57.2	40.5	24.1	83.9	79.6	57.1	0.430	46.6	28.7	14.0	74.8	72.0	47.2	0.428
BM25-Threshold	50.5	38.0	19.5	80.5	78.0	53.3	0.268	41.5	26.2	12.5	73.0	69.0	44.4	0.263
Adaptive-RAG	56.5	40.0	23.0	80.0	77.5	55.4	0.305	45.5	27.0	13.5	73.0	70.0	45.7	0.300
TierMem-2arm	47.6	39.6	20.8	81.8	**81.8**	54.3	0.316	40.9	28.2	12.9	70.4	70.4	44.6	0.314
s3	55.0	39.0	21.5	78.5	75.5	53.9	0.305	45.5	26.5	13.8	71.0	67.0	44.8	0.310
Sysformer	57.5	38.8	21.2	86.0	76.0	55.9	0.285	47.8	23.5	14.2	75.0	69.0	45.9	0.300
Stage 1 (3-arm)	58.3	38.3	20.8	87.0	75.7	56.0	0.272	48.4	23.0	14.5	76.6	70.0	46.5	0.288
Stage 1 (6-arm)	59.0	40.5	23.5	87.2	78.0	57.6	0.255	49.0	28.5	15.5	77.5	71.0	48.3	0.272
FORGE	**60.4**	**42.7**	**26.4**	**87.5**	80.5	**59.5**	0.236	**50.2**	**30.5**	**16.7**	**80.5**	**72.5**	**50.1**	0.249
Oracle (6-arm)	67.4	53.2	34.0	88.5	89.2	66.5	0.168	59.1	42.6	23.2	83.4	82.0	58.1	0.182

4.2 Answer Quality and the Cost of a New Query

Full FORGE reaches 59.5/50.1 macro F1 on Qwen/Mistral, exceeding Always-Raw by 2.4/2.9 points with 45%/42% fewer cached answer tokens (Table 1). It exceeds Sysformer, the strongest adapted baseline here, by 3.6/4.2 points; Table 3 tests a matched lightweight comparator and isolates Stage 2 from action expansion. Fixed Summary leads on Qwen FEVER (81.8 vs. 80.5), which motivates the development-set fallback for selecting a task-specific operating point.

Lite improves Qwen/Mistral F1 by 1.6/1.9 while saving 44%/40% of online tokens with roughly 5.5 ms added p50 latency (Table 2). Full gains accuracy but raises Qwen p50 from 122.5 to 271.4 ms. On the DeepSeek-V3.2 API, zero-shot Lite gives 63.1 F1/0.182k tokens/about 2.2 s versus Raw's 63.4/0.287k/about 2.1 s: 36.6% fewer tokens for 0.3 F1 and a small latency increase. API limits prevented online Full measurement. Lite thus preserves one-call efficiency; Full offers additional

Table 2: **Fresh Online results (batch size 1).** All host calls and input/output tokens are counted; F1/EM are five-task macro scores. Latency is end-to-end, with Full's four feature calls parallelized. Bold marks best quality or lowest online cost per host.

Host	Policy	Host calls		Macro (%)		Online $\bar{C}$ (k)	Latency (ms)	
		Pre	Total	F1	EM		p50	p95
Qwen3-8B	Always-Raw	0	1	57.1	48.6	0.430	122.5	148.9
	FORGE-Lite	0	1	58.7	49.8	**0.242**	128.0	154.1
	Full FORGE	4	5	**59.4**	**50.5**	0.384	271.4	329.8
Mistral-7B	Always-Raw	0	1	47.2	40.5	0.428	129.8	157.7
	FORGE-Lite	0	1	49.1	41.9	**0.256**	135.3	164.8
	Full FORGE	4	5	**50.0**	**42.7**	0.400	291.7	354.6

accuracy when deployment permits the latency of parallel feature probes.

4.3 RELIABILITY, STRONG BASELINES, AND STAGE-2 VALUE

Table 3: **Reliability and matched Stage 2 gains, Cached.** Mean ± SD: three splits × three seeds, equal baseline tuning budgets; 2,500-update rows are single sweep points. Paired-query bootstrap 95% CIs use the canonical split, comparing policies with Raw or Full with six-arm Stage 1 at matched features, decoding, and cost weights. Boxes mark best nine-run means.

Host	Policy	Macro F1 evidence		Cached $\bar{C}$ (k)	Matched Stage 2 effect	
		Nine-run mean ± SD	Δ vs Raw 95% CI		Mean ΔF1	Canonical 95% CI
Qwen3-8B	Always-Raw	57.0 ± 0.4	reference	0.431	–	–
	BGE+BM25-KL	57.5 ± 0.5	$[+0.1, +0.9]$	0.268	–	–
	FORGE-Lite	58.6 ± 0.5	$[+1.1, +2.1]$	0.243	–	–
	Stage 1 (6-arm)	57.6 ± 0.5	–	0.255	reference	–
	Stage 2 (2,500 updates)	58.9	–	0.244	–	–
	Full FORGE (5,000 updates)	$\boxed{\mathbf{59.4}} \pm 0.4$	$[+1.9, +2.9]$	**0.237**	$+1.8$	$[+1.3, +2.3]$
Mistral-7B	Always-Raw	47.1 ± 0.4	reference	0.429	–	–
	BGE+BM25-KL	47.5 ± 0.5	$[0.0, +0.8]$	0.279	–	–
	FORGE-Lite	49.0 ± 0.5	$[+1.4, +2.4]$	0.257	–	–
	Stage 1 (6-arm)	48.3 ± 0.5	–	0.272	reference	–
	Stage 2 (2,500 updates)	49.5	–	0.259	–	–
	Full FORGE (5,000 updates)	$\boxed{\mathbf{50.0}} \pm 0.5$	$[+2.3, +3.5]$	**0.250**	$+1.7$	$[+1.2, +2.2]$

BGE+BM25-KL's Mistral interval starts at zero; the five retuned adaptive baselines have intervals below zero against Raw on both hosts. The canonical Qwen/Mistral ladder separates labels, actions, structure, and host features: hard-label BGE+BM25 56.8/46.7, KL distillation 57.5/47.5, six-arm routing with the same 773 features 58.4/48.6, Lite 58.7/49.1, and Full 59.4/50.0 macro F1.

Table 4: **Cumulative feature ablation.** Qwen3-8B Cached F1 (%); parentheses show changes from the previous row. Full ϕ is the final row, evaluated in a separate feature-study run from Table 1.

Features	HotpotQA	MuSiQue
Structured only	56.8	23.4
+ BGE embedding	60.4 (+3.6)	25.9 (+2.5)
+ Retrieval	59.3 (−1.1)	26.4 (+0.5)
+ Host probe	**61.6** (+2.3)	26.4 (+0.0)
+ Self-consistency	60.6 (−1.0)	**26.9** (+0.5)

Query embeddings yield the largest HotpotQA gain, while the host probe adds 2.3 F1 (Table 4). Retrieval adds four BM25 statistics and one query–passage cosine. Appendix B.5 gives the matched 773-to-789-dimensional comparison.

At matched actions, features, decoding, and cost weights, Stage 2 adds 1.8/1.7 macro F1 on Qwen/Mistral and saves 7.1%/8.1% cached tokens (Table 3); 2,500 updates deliver about 70% of the final gain. Refinement thus improves decisions within the same action space. Training uses $5{,}000 \times 32 \times 8 = 1.28$M host completions, about 384M input/82M output tokens, and 64–96

GPU-hours on one eight-GPU node per operating point. This one-time training cost is separate from recurring feature probes, whose online overhead is measured in Table 2.

4.4 Grounding, Proxy Rewards, and Operating Points

Table 5: **Grounding on Qwen3-8B (%).** Canonical predictions; bold marks the best comparable value for each metric within each task.

Policy	F1	Gold recall	Source supp.	Prompt supp.	Unsup.
HotpotQA					
Raw	57.2	**91.2**	61.9	59.1	30.8
Summary	46.9	73.4	54.0	51.8	37.1
Full	**60.4**	64.8	**65.3**	63.7	**27.4**
FEVER					
Raw	79.6	**94.1**	82.0	80.3	12.9
Summary	**81.8**	87.6	**84.4**	83.1	**10.4**
Full	80.5	79.8	83.7	82.5	10.9

Table 6: **Frozen proxy rewards, Qwen3-8B.** Best outcomes are bold; boxes mark the highest macro F1 and exact match.

Stage 0/1 reward	Stage 2 reward	F1	EM	$\bar{C}$	Source supp.
Gold F1 used in Stage 0/1					
Gold F1	None	57.6	48.9	0.255	65.9
	Gold F1	**59.5**	**50.6**	**0.236**	67.4
	Verifier	58.8	49.8	0.241	66.9
	Judge	59.1	50.1	0.239	**67.5**
No gold F1 in any training stage					
Verifier	Verifier	58.1	49.2	0.248	66.6
Judge	Judge	58.6	49.6	0.245	67.1

Grounding (Table 5). Raw/Summary denote Always-Raw/Always-Summary; Full is Full FORGE. F1 is answer F1; gold recall measures annotated evidence in the prompt. Supp./Unsup. denote support/unsupported rates from a fixed NLI-style evaluator (threshold 0.70). Source support/unsupported rate cover all examples; prompt support uses each policy's Summary/Raw subset and is descriptive, not paired. A balanced 200-example human audit gives 87.0% agreement (Cohen's $\kappa = 0.74$).
Proxy rewards (Table 6). Verifier/Judge denote the frozen verifier/LLM judge; None is Stage 1 only. Canonical Cached evaluation: five-task macro F1/EM use held-out gold labels (%); $\bar{C}$ counts selected-answer k tokens/query. An independent frozen evaluator scores source support (%). Proxy runs reuse hyperparameters selected on gold-labeled development data, as in the gold-reward experiments.

Full improves HotpotQA F1 and lowers unsupported answers: gold evidence recall falls from 91.2 to 64.8 while source support rises from 61.9 to 65.3 (Table 5). Selective evidence delivery can therefore improve answer grounding even with less annotated evidence in the prompt. Fixed Summary leads on all three outcomes for FEVER. These answer-level metrics do not establish reasoning-trace faithfulness, and the generated outputs contain no citations.

A frozen judge supplies all training rewards and reaches 58.6 F1, within 0.9 F1 and 0.3 source-support points of gold-reward training, at 0.245k versus 0.236k cached tokens (Table 6). Gold warm-start with judge-based Stage 2 narrows the F1 gap to 0.4. Proxy rewards thus retain most of the gains; hyperparameter selection still uses labeled development data.

Cost weights $(\lambda_{\text{in}}, \lambda_{\text{out}}) = (0.10, 0.20)$ are tuned on Qwen HotpotQA development data and fixed thereafter. A Fresh Online sweep from $(0.05, 0.00)$ to $(0.20, 0.20)$ yields Qwen Full 59.9 F1/0.410k tokens to 58.7/0.358k, and Lite 59.2/0.268k to 58.0/0.216k. Mistral Full spans 50.5/0.430k to 49.3/0.373k, and Lite 49.6/0.287k to 48.4/0.229k; Full's first point exceeds Raw's 0.428k. A fallback requires a positive lower bound on the paired 95% development utility-difference interval against the best fixed arm. It selects FORGE for HotpotQA/2Wiki/MuSiQue and Summary for PopQA/FEVER: 59.7 F1/50.6 EM/0.235k cached tokens versus Full's 59.5/50.6/0.236k, using labeled development data to select a suitable task-specific operating point.

4.5 Thinking Controls and Transfer

With fixed retrieval and one pre-answer decision, the 12-arm router adds 1.4–1.9 macro F1 over Raw+Think-High at roughly half its cached cost (Table 7). Gains concentrate on multi-hop tasks. Direct+Think-High reaches 38.5/47.3/48.4 F1 versus Raw+Think-High's 63.4/69.6/69.5: evidence still matters at the High budget, motivating joint control of both.

The full policy reaches $D_{\text{KL}} = 0.02$ from the Boltzmann target, versus 0.49 for Stage 1 and 0.64 without the KL anchor (Figure 5). This supports the anchor's role in retaining the target's selective allocation across joint support–thinking combinations.

Table 7: **Joint support and thinking control.** Task F1 (%) and Cached cost (k/query). Six-arm FORGE excludes Think-Low/High; AdaReasoner+ (Wang et al., 2025) fixes support per task. Bold/boxes mark best task/macro F1 within each host.

Host	Policy	Task F1 (%)					Macro F1	$\bar{C}$
		HQA	2Wiki	MSQ	PopQA	FEVER		
Qwen3-8B-Thinking	Raw+NoThink	58.0	41.2	24.5	84.5	80.0	57.6	0.485
	Raw+Think-High	62.5	47.0	30.8	87.0	89.5	63.4	1.280
	Direct+Think-High	38.0	30.5	18.5	38.0	67.5	38.5	0.890
	AdaReasoner+	62.5	46.5	31.0	**88.0**	90.5	63.7	0.952
	FORGE, 6-arm	60.4	42.7	26.4	87.8	82.5	60.0	0.236
	FORGE, 12-arm	**64.8**	**49.2**	**33.5**	**88.0**	**91.0**	**65.3**	0.690
Claude-4-Sonnet	Raw+NoThink	65.5	47.8	30.5	89.5	86.0	63.9	0.482
	Raw+Think-High	70.5	54.0	37.0	91.5	95.0	69.6	1.350
	Direct+Think-High	49.5	36.5	25.5	49.5	75.5	47.3	0.910
	AdaReasoner+	70.0	53.5	36.5	92.5	95.0	69.5	0.985
	FORGE, 6-arm	67.0	49.0	32.0	92.0	88.5	65.7	0.235
	FORGE, 12-arm	**72.4**	**55.5**	**38.8**	**93.0**	**95.5**	**71.0**	0.715
DeepSeek-R1	Raw+NoThink	64.0	46.5	31.5	92.0	86.5	64.1	0.480
	Raw+Think-High	68.5	53.0	38.0	93.5	94.5	69.5	1.420
	Direct+Think-High	50.5	38.0	27.5	50.0	76.0	48.4	0.940
	AdaReasoner+	68.5	53.0	38.0	93.5	95.0	69.6	1.045
	FORGE, 6-arm	65.5	47.5	32.0	93.5	88.0	65.3	0.234
	FORGE, 12-arm	**70.5**	**55.0**	**39.5**	**94.5**	**95.5**	**71.0**	0.730

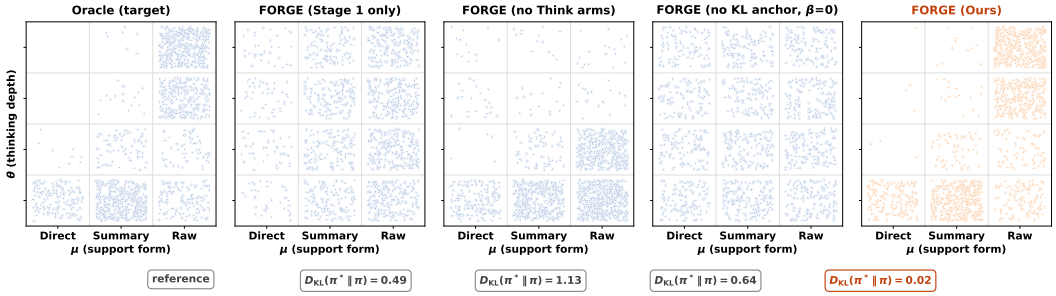

Figure 5: **Joint action distributions** from 1,500 samples per policy. KL compares each distribution with the leftmost Boltzmann training target, distinct from the descriptive Oracle in Tables 1 and S17.

4.6 CROSS-HOST TRANSFER

Table 8: **Zero-shot Cached transfer.** A Qwen-trained Full router with no target-host Stage 2. Cost excludes fresh probes. Bold marks best F1/cost per host; Appendix C.5 gives per-task results.

Target host	Macro F1 (%)		Cached $\bar{C}$ (k)	
	Raw	FORGE	Raw	FORGE
Llama-3.3-70B	58.5	**61.2**	0.318	**0.205**
DeepSeek-V3.2	**63.4**	**63.4**	0.287	**0.176**
Qwen3.5-397B	63.1	**65.6**	0.305	**0.198**

Full matches or exceeds Raw in 11/15 task–host cells, with macro F1 gains of +2.7/+0.0/+2.5 and 35.5%/38.7%/35.1% fewer cached tokens (Table 8). One source-trained router improves the quality–cost balance across host families and scales; DeepSeek ties Raw at the reported precision with the largest token saving. The online Lite result supports no-probe API use. Target-host calibration could further tailor Direct support to each host's prior knowledge.

5 CONCLUSION

FORGE jointly controls evidence form and reasoning depth around a frozen host. Its factorized router distills an entropy-regularized quality–cost target and refines decisions through sampled host feedback. Across five tasks, Full improves nine-run cached macro F1 over Raw by 2.4/2.9 points on Qwen/Mistral; matched controls attribute 1.8/1.7 points to Stage 2. Lite improves local-host F1 and token cost with one online call, while Full uses four parallel probes for further accuracy. FEVER favors Summary; DeepSeek transfer primarily saves tokens. Results across host families establish joint evidence-form and thinking control as a practical adaptation mechanism with explicit quality, token, and latency tradeoffs across host families and deployment settings.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Pranjal Aggarwal, Aman Madaan, Ankit Anand, Srividya Pranavi Potharaju, Swaroop Mishra, Pei Zhou, Aditya Gupta, Dheeraj Rajagopal, Karthik Kappaganthu, Yiming Yang, Shyam Upadhyay, Manaal Faruqui, and Mausam. Automix: Automatically mixing language models. *Advances in Neural Information Processing Systems*, 37:131000–131034, 2024.

Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce-style optimization for learning from human feedback in llms. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12248–12267, 2024.

Mohammad Ali Alomrani, Yingxue Zhang, Derek Li, Qianyi Sun, Soumyasundar Pal, Zhanguang Zhang, Yaochen Hu, Rohan Deepak Ajwani, Antonios Valkanas, Raika Karimi, Peng Cheng, Yunzhou Wang, Pengyi Liao, Hanrui Huang, Bin Wang, Jianye Hao, and Mark Coates. Reasoning on a budget: A survey of adaptive and controllable test-time compute in llms. *arXiv preprint arXiv:2507.02076*, 2025.

Anthropic. Claude 3.7 sonnet and claude code. `https://www.anthropic.com/news/claude-3-7-sonnet`, 2025. Blog post.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avi Sil, and Hannaneh Hajishirzi. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, pp. 9112–9141, 2024.

Parth Asawa, Alan Zhu, Matei Zaharia, Alex Dimakis, and Joseph E Gonzalez. How to train your advisor: Steering black-box llms with advisor models. In *First Workshop on Foundations of Reasoning in Language Models*, 2025.

Prakhar Bansal and Shivangi Agarwal. Lightweight query routing for adaptive rag: A baseline study on ragrouter-bench. *arXiv preprint arXiv:2604.03455*, 2026.

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*, 2024.

Jiale Cheng, Xiao Liu, Kehan Zheng, Pei Ke, Hongning Wang, Yuxiao Dong, Jie Tang, and Minlie Huang. Black-box prompt optimization: Aligning large language models without model training. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3201–3219, 2024a.

Xin Cheng, Xun Wang, Xingxing Zhang, Tao Ge, Si-Qing Chen, Furu Wei, Huishuai Zhang, and Dongyan Zhao. xrag: Extreme context compression for retrieval-augmented generation with one token. *Advances in Neural Information Processing Systems*, 37:109487–109516, 2024b.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks VS Lakshmanan, and Ahmed Hassan Awadallah. Hybrid llm: Cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations*, 2024.

Dujian Ding, Ankur Mallick, Shaokun Zhang, Chi Wang, Daniel Madrigal, Mirian Del Carmen Hipolito Garcia, Menglin Xia, Laks VS Lakshmanan, Qingyun Wu, and Victor Rühle. Best-route: Adaptive llm routing with test-time optimal compute. In *International Conference on Machine Learning*, pp. 13870–13884. PMLR, 2025.

Mohamed Amine Ferrag, Norbert Tihanyi, and Merouane Debbah. From llm reasoning to autonomous ai agents: A comprehensive review. *arXiv preprint arXiv:2504.19678*, 2025.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*, 2(1):32, 2023.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, Xiaokang Zhang, Xingkai Yu, Yu Wu, Z. F. Wu, Zhibin Gou, Zhihong Shao, Zhuoshu Li, Ziyi Gao, Aixin Liu, Bing Xue, Bingxuan Wang, Bochao Wu, Bei Feng, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Hanwei Xu, Honghui Ding, Huazuo Gao, Hui Qu, Hui Li, Jianzhong Guo, Jiashi Li, Jingchang Chen, Jingyang Yuan, Jinhao Tu, Junjie Qiu, Junlong Li, J. L. Cai, Jiaqi Ni, Jian Liang, Jin Chen, Kai Dong, Kai Hu, Kaichao You, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Liang Zhao, Litong Wang, Liyue Zhang, Lei Xu, Leyi Xia, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingxu Zhou, Meng Li, Miaojun Wang, Mingming Li, Ning Tian, Panpan Huang, Peng Zhang, Qiancheng Wang, Qinyu Chen, Qiushi Du, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, R. J. Chen, R. L. Jin, Ruyi Chen, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shengfeng Ye, Shiyu Wang, Shuiping Yu, Shunfeng Zhou, Shuting Pan, S. S. Li, Shuang Zhou, Shaoqing Wu, Tao Yun, Tian Pei, Tianyu Sun, T. Wang, Wangding Zeng, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, W. L. Xiao, Wei An, Xiaodong Liu, Xiaohan Wang, Xiaokang Chen, Xiaotao Nie, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, X. Q. Li, Xiangyue Jin, Xiaojin Shen, Xiaosha Chen, Xiaowen Sun, Xiaoxiang Wang, Xinnan Song, Xinyi Zhou, Xianzu Wang, Xinxia Shan, Y. K. Li, Y. Q. Wang, Y. X. Wei, Yang Zhang, Yanhong Xu, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Wang, Yi Yu, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yuan Ou, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yunfan Xiong, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Y. X. Zhu, Yanping Huang, Yaohui Li, Yi Zheng, Yuchen Zhu, Yunxian Ma, Ying Tang, Yukun Zha, Yuting Yan, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhicheng Ma, Zhigang Yan, Zhiyu Wu, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Zizheng Pan, Zhen Huang, Zhipeng Xu, Zhongyu Zhang, and Zhen Zhang. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pp. 1861–1870. Pmlr, 2018.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop qa dataset for comprehensive evaluation of reasoning steps. In *Proceedings of the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

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 *International Conference on Learning Representations*, 2022.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. Routerbench: A benchmark for multi-llm routing system. In *Agentic Markets Workshop at ICML 2024*, 2024.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 7036–7050, 2024.

Pengcheng Jiang, Jiacheng Lin, Zhiyi Shi, Zifeng Wang, Luxi He, Yichen Wu, Ming Zhong, Peiyang Song, Qizheng Zhang, Heng Wang, Xueqiang Xu, Hanwen Xu, Pengrui Han, Dylan Zhang, Jiashuo Sun, Chaoqi Yang, Kun Qian, Tian Wang, Changran Hu, Manling Li, Quanzheng Li, Hao Peng,

Sheng Wang, Jingbo Shang, Chao Zhang, Jiaxuan You, Liyuan Liu, Pan Lu, Yu Zhang, Heng Ji, Yejin Choi, Dawn Song, Jimeng Sun, and Jiawei Han. Adaptation of agentic ai. *arXiv preprint arXiv:2512.16301*, 2025a.

Pengcheng Jiang, Xueqiang Xu, Jiacheng Lin, Jinfeng Xiao, Zifeng Wang, Jimeng Sun, and Jiawei Han. s3: You don't need that much data to train a search agent via rl. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 21610–21628, 2025b.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Hailey Joren, Jianyi Zhang, Chun-Sung Ferng, Da-Cheng Juan, Ankur Taly, and Cyrus Rashtchian. Sufficient context: A new lens on retrieval augmented generation systems. In *The Thirteenth International Conference on Learning Representations*, 2025.

Minki Kang, Wei-Ning Chen, Dongge Han, Huseyin A Inan, Lukas Wutschitz, Yanzhi Chen, Robert Sim, and Saravan Rajmohan. Acon: Optimizing context compression for long-horizon llm agents. *arXiv preprint arXiv:2510.00615*, 2025.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 6769–6781, 2020.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020a.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020b.

Ruoran Li, Xinghua Zhang, Haiyang Yu, Shitong Duan, Xiang Li, Wenxin Xiang, Chonghua Liao, Xudong Guo, Yongbin Li, and Jinli Suo. Mempo: Self-memory policy optimization for long-horizon agents. *arXiv preprint arXiv:2603.00680*, 2026.

Zhuowan Li, Cheng Li, Mingyang Zhang, Qiaozhu Mei, and Michael Bendersky. Retrieval augmented generation or long-context llms? a comprehensive study and hybrid approach. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 881–893, 2024.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective. In *Second Conference on Language Modeling*, 2025.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. Evaluating very long-term conversational memory of llm agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13851–13870, 2024.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9802–9822. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.acl-long.546. URL https://aclanthology.org/2023.acl-long.546/.

D MCFADDEN. Conditional logit analysis of qualitative choice behavior. *Frontier in Econometrics*, 1973.

Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 20286–20332, 2025.

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 from preference data. In *International Conference on Learning Representations*, 2025.

OpenAI, :, Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, Alex Iftimie, Alex Karpenko, Alex Tachard Passos, Alexander Neitz, Alexander Prokofiev, Alexander Wei, Allison Tam, Ally Bennett, Ananya Kumar, Andre Saraiva, Andrea Vallone, Andrew Duberstein, Andrew Kondrich, Andrey Mishchenko, Andy Applebaum, Angela Jiang, Ashvin Nair, Barret Zoph, Behrooz Ghorbani, Bohan Zhang, Ben Rossen, Benjamin Sokolowsky, Boaz Barak, Bob McGrew, Borys Minaiev, Botao Hao, Bowen Baker, Brandon Houghton, Brandon McKinzie, Brydon Eastman, Camillo Lugaresi, Cary Bassin, Cary Hudson, Chak Ming Li, Charles de Bourcy, Chelsea Voss, Chen Shen, Chong Zhang, Chris Koch, Chris Orsinger, Christopher Hesse, Claudia Fischer, Clive Chan, Dan Roberts, Daniel Kappler, Daniel Levy, Daniel Selsam, David Dohan, David Farhi, David Mely, David Robinson, Dimitris Tsipras, Doug Li, Dragos Oprica, Eben Freeman, Eddie Zhang, Edmund Wong, Elizabeth Proehl, Enoch Cheung, Eric Mitchell, Eric Wallace, Erik Ritter, Evan Mays, Fan Wang, Felipe Petroski Such, Filippo Raso, Florencia Leoni, Foivos Tsimpourlas, Francis Song, Fred von Lohmann, Freddie Sulit, Geoff Salmon, Giambattista Parascandolo, Gildas Chabot, Grace Zhao, Greg Brockman, Guillaume Leclerc, Hadi Salman, Haiming Bao, Hao Sheng, Hart Andrin, Hessam Bagherinezhad, Hongyu Ren, Hunter Lightman, Hyung Won Chung, Ian Kivlichan, Ian O'Connell, Ian Osband, Ignasi Clavera Gilaberte, Ilge Akkaya, Ilya Kostrikov, Ilya Sutskever, Irina Kofman, Jakub Pachocki, James Lennon, Jason Wei, Jean Harb, Jerry Twore, Jiacheng Feng, Jiahui Yu, Jiayi Weng, Jie Tang, Jieqi Yu, Joaquin Quiñonero Candela, Joe Palermo, Joel Parish, Johannes Heidecke, John Hallman, John Rizzo, Jonathan Gordon, Jonathan Uesato, Jonathan Ward, Joost Huizinga, Julie Wang, Kai Chen, Kai Xiao, Karan Singhal, Karina Nguyen, Karl Cobbe, Katy Shi, Kayla Wood, Kendra Rimbach, Keren Gu-Lemberg, Kevin Liu, Kevin Lu, Kevin Stone, Kevin Yu, Lama Ahmad, Lauren Yang, Leo Liu, Leon Maksin, Leyton Ho, Liam Fedus, Lilian Weng, Linden Li, Lindsay McCallum, Lindsey Held, Lorenz Kuhn, Lukas Kondraciuk, Lukasz Kaiser, Luke Metz, Madelaine Boyd, Maja Trebacz, Manas Joglekar, Mark Chen, Marko Tintor, Mason Meyer, Matt Jones, Matt Kaufer, Max Schwarzer, Meghan Shah, Mehmet Yatbaz, Melody Y. Guan, Mengyuan Xu, Mengyuan Yan, Mia Glaese, Mianna Chen, Michael Lampe, Michael Malek, Michele Wang, Michelle Fradin, Mike McClay, Mikhail Pavlov, Miles Wang, Mingxuan Wang, Mira Murati, Mo Bavarian, Mostafa Rohaninejad, Nat McAleese, Neil Chowdhury, Neil Chowdhury, Nick Ryder, Nikolas Tezak, Noam Brown, Ofir Nachum, Oleg Boiko, Oleg Murk, Olivia Watkins, Patrick Chao, Paul Ashbourne, Pavel Izmailov, Peter Zhokhov, Rachel Dias, Rahul Arora, Randall Lin, Rapha Gontijo Lopes, Raz Gaon, Reah Miyara, Reimar Leike, Renny Hwang, Rhythm Garg, Robin Brown, Roshan James, Rui Shu, Ryan Cheu, Ryan Greene, Saachi Jain, Sam Altman, Sam Toizer, Sam Toyer, Samuel Miserendino, Sandhini Agarwal, Santiago Hernandez, Sasha Baker, Scott McKinney, Scottie Yan, Shengjia Zhao, Shengli Hu, Shibani Santurkar, Shraman Ray Chaudhuri, Shuyuan Zhang, Siyuan Fu, Spencer Papay, Steph Lin, Suchir Balaji, Suvansh Sanjeev, Szymon Sidor, Tal Broda, Aidan Clark, Tao Wang, Taylor Gordon, Ted Sanders, Tejal Patwardhan, Thibault Sottiaux, Thomas Degry, Thomas Dimson, Tianhao Zheng, Timur Garipov, Tom Stasi, Trapit Bansal, Trevor Creech, Troy Peterson, Tyna Eloundou, Valerie Qi, Vineet Kosaraju, Vinnie Monaco, Vitchyr Pong, Vlad Fomenko, Weiyi Zheng, Wenda Zhou, Wenting Zhan, Wes McCabe, Wojciech Zaremba, Yann Dubois, Yinghai Lu, Yining Chen, Young Cha, Yu Bai, Yuchen He, Yuchen Zhang, Yunyun Wang, Zheng Shao, and Zhuohan Li. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 5687–5711, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in neural information processing systems*, 36:53728–53741, 2023.

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. *Advances in neural information processing systems*, 30, 2017.

Ranjan Sapkota, Konstantinos I Roumeliotis, and Manoj Karkee. Ai agents vs. agentic ai: A conceptual taxonomy, applications and challenges. *Information Fusion*, pp. 103599, 2025.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Kartik Sharma, Yiqiao Jin, Vineeth Rakesh, Yingtong Dou, Menghai Pan, Mahashweta Das, and Srijan Kumar. Sysformer: Safeguarding frozen large language models with adaptive system prompts. In *ICML 2025 Workshop on Reliable and Responsible Foundation Models*, 2025.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 8371–8384, 2024.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Weihang Su, Yichen Tang, Qingyao Ai, Zhijing Wu, and Yiqun Liu. Dragin: Dynamic retrieval augmented generation based on the real-time information needs of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12991–13013, 2024.

Tianxiang Sun, Yunfan Shao, Hong Qian, Xuanjing Huang, and Xipeng Qiu. Black-box tuning for language-model-as-a-service. In *International Conference on Machine Learning*, pp. 20841–20855. PMLR, 2022.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multi-hop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pp. 10014–10037, 2023.

Prince Zizhuang Wang and Shuli Jiang. Prime: Training free proactive reasoning via iterative memory evolution for user-centric agent. *arXiv preprint arXiv:2604.07645*, 2026.

Xiangqi Wang, Yue Huang, Yanbo Wang, Xiaonan Luo, Kehan Guo, Yujun Zhou, and Xiangliang Zhang. Adareasoner: Adaptive reasoning enables more flexible thinking in large language models. *arXiv preprint arXiv:2505.17312*, 2025.

Ziqi Wang, Xi Zhu, Shuhang Lin, Haochen Xue, Minghao Guo, and Yongfeng Zhang. Ragrouter-bench: A dataset and benchmark for adaptive rag routing. *arXiv preprint arXiv:2602.00296*, 2026.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. From decoding to meta-generation: Inference-time algorithms for large language models. *Transactions on Machine Learning Research*, 2024.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pp. 641–649, 2024.

Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. Corrective retrieval augmented generation. *arXiv preprint arXiv:2401.15884*, 2024.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question

answering. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Linan Yue, Yichao Du, Yizhi Wang, Weibo Gao, Fangzhou Yao, Li Wang, Ye Liu, Ziyu Xu, Qi Liu, Shimin Di, and Min-Ling Zhang. Don't overthink it: A survey of efficient r1-style large reasoning models. *arXiv preprint arXiv:2508.02120*, 2025.

Heng Zhou, Zelin Tan, Zhemeng Zhang, Yutao Fan, Yibing Lin, Li Kang, Xiufeng Song, Rui Li, Songtao Huang, Ao Yu, Yuchen Fan, Yanxu Chen, Kaixin Xu, Xiaohong Liu, Yiran Qin, Philip Torr, Chen Zhang, and Zhenfei Yin. Select-then-solve: Paradigm routing as inference-time optimization for llm agents. *arXiv preprint arXiv:2604.06753*, 2026.

Huichi Zhou, Yihang Chen, Siyuan Guo, Xue Yan, Kin Hei Lee, Zihan Wang, Ka Yiu Lee, Guchun Zhang, Kun Shao, Linyi Yang, and Jun Wang. Memento: Fine-tuning llm agents without fine-tuning llms. *arXiv preprint arXiv:2508.16153*, 2025.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. From lossy to verified: A provenance-aware tiered memory for agents. In *ICLR 2026 Workshop on Memory for LLM-Based Agentic Systems*, 2026.

Brian D Ziebart, Andrew Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *Proceedings of the 23rd national conference on Artificial intelligence-Volume 3*, pp. 1433–1438, 2008.

Technical Appendices

A Related Work

Section 1 framed single-step, pre-answer routing around a frozen language-model host as a design space with one intrinsic axis (how deeply the model reasons, when the host exposes such a control) and three extrinsic axes (which model to call, how aggressively to retrieve, and what form the retrieved support takes). The three subsections below organize prior work along this taxonomy. Section A.1 covers intrinsic adaptive thinking, Section A.2 covers the two extrinsic axes that have received sustained attention, and Section A.3 covers the support-form axis that FORGE completes.

A.1 Intrinsic Adaptive Thinking

A first line of work allocates inference compute inside the backbone itself. Frontier reasoning systems learn to expend more decoding tokens on harder queries through chain-of-thought training (OpenAI et al., 2024; Guo et al., 2025), and recent assistants expose this as a user-facing thinking budget that is set per request (Anthropic, 2025). A growing body of work studies how to control this budget automatically rather than by user toggle. Muennighoff et al. (2025) show that a simple budget-forcing trick at test time recovers most of the gains of full reasoning training on a small base model. Wang et al. (2025) train an RL controller that selects reasoning configurations (temperature, depth) per query for any backbone, with theoretical convergence guarantees. The recent survey of Alomrani et al. (2025) catalogs this space along an L1 (controllability) versus L2 (adaptiveness) axis and lists more than a dozen budget controllers built on top of reasoning-trained hosts.

All of these methods presuppose either a reasoning-trained backbone or a user-exposed thinking toggle, and they condition behavior on query difficulty by changing how the model itself executes. Frozen hosts deployed behind closed APIs or pinned checkpoints cannot generally be retrained by the application, although some expose a per-request reasoning budget. FORGE routes support form and exposed thinking options around a fixed host; our evaluation covers single-step decisions.

A.2 Extrinsic Adaptivity: Model and Retrieval Routing

The first extrinsic axis routes queries among backbones of different cost and capability. FrugalGPT (Chen et al., 2024) cascades a query through increasingly expensive models and stops when a verifier reports sufficient confidence. RouteLLM (Ong et al., 2025) trains a preference-based router that selects a strong or weak model per query. AutoMix (Aggarwal et al., 2024) adds self-verification to estimate answer reliability before escalation. BEST-Route (Ding et al., 2025) jointly routes among models and response sample counts under an explicit cost-quality Pareto objective, the closest prior work in spirit to our framing. These methods also study cost–quality routing, but their decision variable is the backbone itself. FORGE fixes the host and routes the support form for each query.

The second extrinsic axis decides whether and how aggressively to retrieve. Retrieval-augmented generation prepends retrieved passages to every query (Lewis et al., 2020a; Shi et al., 2024); adaptive variants relax this default along several sub-axes. Self-RAG (Asai et al., 2024) trains the language model to emit reflection tokens that trigger retrieval on demand, at the cost of modifying the host. CRAG (Yan et al., 2024) adds a post-hoc evaluator that triggers corrective retrieval when document quality is low, and IRCoT (Trivedi et al., 2023) interleaves retrieval steps with chain-of-thought reasoning for multi-hop queries. Adaptive-RAG (Jeong et al., 2024) trains a query-complexity classifier to pick among no retrieval, single-step retrieval, and iterative multi-step retrieval. Self-Route (Li et al., 2024) frames the decision as a per-query binary choice between RAG and long-context generation. More recent routing benchmarks (Wang et al., 2026; Bansal & Agarwal, 2026; Zhou et al., 2026) catalog inference-time strategies such as Direct, CoT, and ReAct, and study how to route among them. The *sufficient context* lens of Joren et al. (2025) treats sufficiency as a property of query–context pairs and uses it to gate abstention. Its analysis of frozen hosts motivates treating support form as a separate decision about evidence delivery.

Both axes operate on *whether* or *how deeply* to retrieve. Retrieved content typically reaches the host as raw passages or a near-equivalent, leaving support form fixed. Adaptive-RAG (Jeong et al., 2024) is evaluated in Section 4, and a host-confidence Self-Routing heuristic is reported in Appendix C.8.

A.3 Support-Form Routing with a Fixed Host

The third extrinsic axis selects what *form* the external support takes, holding the backbone and the retrieval backend fixed. Prior tiered-memory systems already route between two support forms within memory hierarchies. TierMem (Zhu et al., 2026) organizes memory into a two-tier hierarchy of summaries and raw pages with a sufficiency router that escalates from summary to raw, reducing tokens by 54% on LoCoMo (Maharana et al., 2024) with minimal accuracy loss. MemPO (Li et al., 2026) and Memento (Zhou et al., 2025) apply reinforcement learning to optimize memory management, and PRIME (Wang & Jiang, 2026) builds experience libraries through iterative evolution without backbone training. A different angle is offered by advisor models (Asawa et al., 2025), which train a small policy that emits free-form natural-language steering instructions to a black-box host on a per-instance basis. TierMem routes between summary and raw but does not include the Direct option that bypasses retrieved support; advisor models deliver free-form hints rather than choosing among the fixed Direct, Summary, and Raw support forms studied here.

Jiang et al. (2025a) catalog the space of agent-supervised tool adaptation, in which a frozen agent supervises the training of a small downstream module via reward feedback, and place tiered-memory routers, advisor models, and similar systems in this family. Two additional architectural precedents matter for FORGE's design, even though each was originally motivated by a different task. Jiang et al. (2025b) introduce *s3*, a PPO-trained T2 search agent that decouples the retriever from the generator and optimizes a Gain-Beyond-RAG reward; s3's choice of on-policy PPO over a frozen host establishes that small reward-based routers can train on frontier-scale hosts, and we include it as a learned-routing baseline in Section 4. Sharma et al. (2025) introduce *Sysformer*, a transformer-based adapter that updates system prompts for frozen LLMs in the *safety* setting (refusal of harmful prompts), and establish that attention-based adapters can steer frozen hosts without weight access; we retarget their adapter template to our support-form alphabet as a transformer-based learned-routing baseline, acknowledging that this retargeting is a partial repurposing rather than a like-for-like comparison. FORGE studies a three-way Direct/Summary/Raw support choice around a frozen host, factorizes it with an optional thinking-depth choice (Section 2.3), and refines the routing policy against measured host feedback. Its distinction from TierMem is the additional Direct arm and joint decision studied under the stated single-step task protocol, not the invention of support-form routing itself. TierMem-style two-arm routing, s3's PPO-trained searcher, and the retargeted Sysformer adapter are all included as experimental baselines in Section 4.

The three forms differ in prompt length and evidence content. FORGE compares their measured answer quality and token use under the specified utility; no information-bottleneck or rate-distortion optimality claim is needed for this empirical choice. A complementary line in agent context compression (Kang et al., 2025) condenses observation histories for long-horizon agents. FORGE studies the narrower, single-answer decision and does not evaluate long-horizon interactions.

B Implementation Details

All experiments were conducted on an HPC cluster (details anonymized for double-blind review). Local open-source hosts were served on 64 GB HBM GPU accelerators, with up to **128 accelerators running concurrently** across 16 nodes at peak. Closed-source and API-served hosts were queried through the corresponding managed inference endpoints. The final experimental matrix spans **8 LLM backbones** ranging from 7B to 671B parameters, **5 benchmarks**, and **40 host–benchmark cells**. Stage 1 supervised training is CPU-only and completes in under two minutes; Stage 2 GRPO refinement runs on a single 8-GPU node and completes in 8 to 12 hours per Pareto operating point.

B.1 Hardware and Software Environment

Table S1 lists the hardware and software used in our experiments.

B.2 Frozen Host Configuration

The host pool is partitioned into three groups by deployment mode and intrinsic capability (Table S2). Locally hosted models are loaded in float16 and sharded across eight HBM accelerators per node via `device_map="auto"`. Qwen3-8B-Instruct and Qwen3-8B-Thinking denote the same Qwen3-8B

Table S1: Hardware and software environment.

Category	Value
Cluster	
System	HPC cluster (anonymized for review)
Node CPU	64-core x86_64
Node GPUs	8 × 64 GB HBM accelerators
Node memory	512 GB
Software stack	
OS	Linux
GPU runtime	6.2.4 (vendor-specific)
Python	3.11.11
PyTorch	2.6.0 (with vendor-specific GPU backend)
Transformers	5.0.0
Sentence-Transformers	5.4.1
Datasets (HF)	4.0.0
scikit-learn	1.6.1
rank_bm25	0.2.2

checkpoint run with its thinking mode disabled and enabled, respectively. API models are accessed through managed inference endpoints without weight access. The cross-host API tables use cached-feature evaluation; only Lite has a measured Fresh Online API example, and Full FORGE's Fresh Online API latency is unmeasured.

Table S2: Frozen LLM backbones grouped by deployment mode and thinking capability.

Host	Family	Parameters	Precision	Deployment
Non-thinking local hosts (main results, Table 1)				
Qwen3-8B-Instruct (main)	Qwen 3	8B	float16	Local (8 accelerators)
Mistral-7B-Instruct-v0.3	Mistral	7B	float16	Local (8 accelerators)
Thinking-capable hosts (composition experiment, Table 7)				
Qwen3-8B-Thinking	Qwen 3	8B	float16	Local (8 accelerators)
Claude Sonnet 4	Anthropic	–	–	API
DeepSeek-R1	DeepSeek	671B MoE	–	API
Frontier non-thinking hosts (cross-host transfer, Table 8)				
Llama-3.3-70B-Instruct	Llama 3.3	70B	–	API
DeepSeek-V3.2	DeepSeek	671B MoE	–	API
Qwen3.5-397B-A17B	Qwen 3.5	397B MoE	–	API

Table S3 lists the decoding parameters. All non-thinking arms share identical decoding to isolate the effect of the action from decoding variability. Self-consistency (SC) probes use temperature sampling with a short per-call budget; the full online cost still includes all three samples and the greedy probe. Think-Low and Think-High set the host's per-request reasoning budget (construction in Appendix B.3).

Table S3: Decoding and inference parameters across host groups.

Parameter	Local hosts	API hosts
`max_new_tokens` (arm answer)	64	64
`max_new_tokens` (SC sample)	24	24
Greedy temperature	0	0
SC temperature	0.7	0.7
SC top-p	0.9	0.9
SC samples K_{sc}	3	3
Think-Low budget (output tokens)	1,024	1,024
Think-High budget (output tokens)	4,096	4,096
API rate limit used	–	25 requests/min

B.3 Retrieval and Action Construction

All actions share a fixed retrieval backend so that performance differences reflect routing rather than retriever tuning (Table S4). Each action $a = (\mu, \theta)$ selects support form μ and thinking setting θ.

Summary form construction. The Summary action is constructed deterministically without any auxiliary language model: BM25 retrieves the top-3 passages, all sentences within those passages are re-ranked by BM25 against the query, and the top-ranked sentences are concatenated until a budget of $\sim$140 words is reached. No second-stage LLM call or learned compressor is invoked for Summary, so its selected-answer input tokens are its host prompt length. This statement concerns support construction; Full FORGE's pre-routing host probes are accounted for separately under the fresh-online protocol in Appendix B.9. We use fixed BM25 and deterministic extractive Summary. Learned compressors and other retrieval backends require separate training and evaluation.

Thinking-setting construction. NoThink uses plain decoding without a thinking instruction, and CoT-Prompt appends the suffix in Table S4; both settings are available on every host. On thinking-capable hosts, Think-Low and Think-High set the host's per-request reasoning budget to $1{,}024$ and $4{,}096$ output tokens, respectively; all other decoding parameters follow Table S3.

Table S4: Retrieval pipeline and action construction.

Component	**Value**
Retriever	BM25 (`rank_bm25`)
Embedding model	BAAI/bge-base-en-v1.5 (768-d, normalized)
Tokenizer	lowercase + punctuation removal
Stopword list	57 common English stopwords
Top-k for Summary/Raw	3 passages
Support-form axis $\mathcal{M}$	
Direct ($\mu = 0$)	Question only, 0 extra input tokens
Summary ($\mu = 1$)	Query-aware sentence selection; $\leq$140 words
Raw ($\mu = 2$)	Top-3 BM25 passages; $\leq$200 words per passage
Thinking-depth axis Θ (thinking-capable hosts)	
NoThink	Plain decoding; no thinking instruction
CoT-Prompt	"Let us think step by step" suffix
Think-Low	Reasoning budget $1{,}024$ output tokens
Think-High	Reasoning budget $4{,}096$ output tokens

B.4 Router Architecture and Three-Stage Training

Full FORGE uses a factorized MLP with a $789 \to 256 \to 256$ shared encoder, followed by support-form and thinking-depth heads (Section 3.1). FORGE-Lite retrains the router on the 778 host-independent input dimensions. Training proceeds in three stages: offline arm enumeration on the warm-start subset (Stage 0), supervised KL distillation against the Boltzmann target (Stage 1), and policy-guided group-relative refinement using host feedback on the expanded action alphabet (Stage 2). Table S5 lists all router-specific hyperparameters.

Stage-1 warm-start objective (full form). The warm-start router π_{θ_0} is trained by minimizing the KL divergence on the μ head against the Boltzmann target π^*_{Boltz} restricted to $\mathcal{A}_{\mathrm{warm}}$ at temperature τ_0=1.0:

$$\mathcal{L}_{\mathrm{warm}}(\theta) \;=\; \frac{1}{N_{\mathrm{warm}}} \sum_i D_{\mathrm{KL}}\big(\pi^*_{\mathrm{Boltz}}(\cdot \mid x_i) \,\big\|\, \pi^{\mu}_{\theta}(\cdot \mid h(x_i))\big), \tag{5}$$

with π^{θ}_{θ} initialized uniform. The target is the closed-form optimum of the stated entropy-regularized finite-action utility on $\mathcal{A}_{\mathrm{warm}}$; the finite-capacity Stage-1 router only approximates this target. The default warm-start uses three actions. The controlled Stage-2 comparison also evaluates an additional full-alphabet, six-action Stage-1 control under matched features, decoding, and cost weights; that control is distinct from the three-action warm-start row in the canonical table.

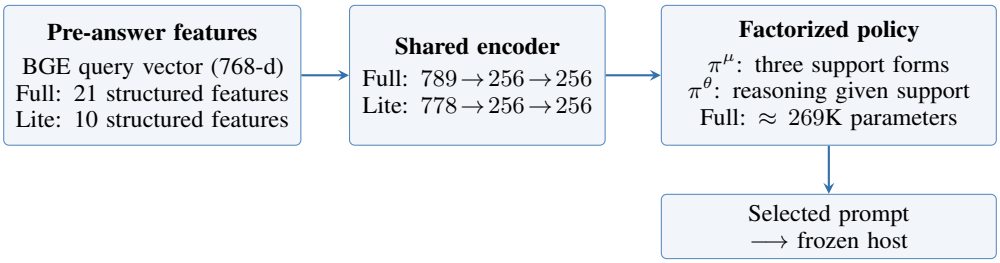

Figure S1: **Full and Lite router deployment paths.** Full uses four pre-routing host generations to construct its 11 host-derived features on a fresh query; Lite is retrained without them. Both variants select one joint action before the frozen host answers, and the host weights remain fixed.

Stage-2 clipped policy and auxiliary losses (full form). Let $\rho_{b,g}(\theta) = \pi_\theta(a_{b,g} \mid x_b)/\pi_{\theta_{\text{old}}}(a_{b,g} \mid x_b)$ be the displayed old-policy ratio. The clipped surrogate uses the PPO-style construction of Schulman et al. (2017) with the group-relative advantage $\hat{A}_{b,g}$ of Shao et al. (2024) (Eq. 4):

$$\mathcal{L}_{\text{pol}}(\theta) \;=\; -\frac{1}{BG}\sum_{b,g}\min\Big(\rho_{b,g}\,\hat{A}_{b,g},\;\; \text{clip}(\rho_{b,g}, 1-\eta, 1+\eta)\,\hat{A}_{b,g}\Big), \tag{6}$$

with $\eta = 0.2$. The action groups are stratified to cover support forms when $G \geq |\mathcal{M}|$ (Table S5). This changes the behavior proposal to a distribution $q(a \mid x)$ that need not equal $\pi_{\theta_{\text{old}}}(a \mid x)$. Without a proposal correction involving q, the displayed ratio does not make Eq. 6 an unbiased policy-gradient estimator for the old policy. We therefore treat Stage 2 as a policy-guided, group-relative clipped surrogate aligned with measured utility, not an exact gradient step toward or convergence guarantee for the Boltzmann target. The total Stage-2 loss is $\mathcal{L}_{\text{GRPO}} = \mathcal{L}_{\text{pol}} + \mathcal{L}_{\text{KL}} + \mathcal{L}_{\text{ent}}$ with the auxiliary terms

$$\mathcal{L}_{\text{KL}}(\theta) \;=\; \beta \cdot \frac{1}{B}\sum_{b} D_{\text{KL}}\big(\pi_\theta(\cdot \mid x_b) \,\big\|\, \pi_{\theta_0}(\cdot \mid x_b)\big), \tag{7}$$

$$\mathcal{L}_{\text{ent}}(\theta) \;=\; -\alpha \cdot \frac{1}{B}\sum_{b} H\big(\pi_\theta(\cdot \mid x_b)\big). \tag{8}$$

The KL anchor keeps the clipped update close to the Stage-1 warm-start reference π_{θ_0}; the entropy bonus discourages premature collapse on the larger alphabet. Optimization uses AdamW at learning rate 1×10^{-5} with $K_{\text{inner}} = 4$ minibatch updates per rollout before $\pi_{\theta_{\text{old}}}$ is refreshed. The KL coefficient β follows an adaptive schedule: it is multiplied by 1.5 every 100 steps when the observed per-query KL exceeds 0.05, and divided by the same factor when it falls below 0.005, so that the policy neither drifts off the warm-start manifold nor stagnates on it. Stage 0 costs $|\mathcal{A}_{\text{warm}}| \cdot N_{\text{warm}} \approx 3 \cdot 1500 = 4500$ host calls (single node, under one hour); Stage 1 is CPU-only and completes in under two minutes. Each Stage-2 operating point uses $T_{\text{grpo}}BG = 5000 \times 32 \times 8 = 1{,}280{,}000$ atomic model completions, about 384M input and 82M output tokens in the measured resource accounting, and 8–12 hours on one 8-GPU node (64–96 GPU-hours). BGE and MLP routing take 0.5–5.5 ms in the measured settings, excluding host calls that acquire Full FORGE's probes on unseen queries. Appendix B.9 includes those calls in Fresh Online token and latency accounting.

B.5 Feature Extraction

Full FORGE's input $\phi_{\text{Full}}(x) \in \mathbb{R}^{789}$ combines six feature groups (Table S6). Eleven dimensions depend on a greedy host probe and three self-consistency samples, so an unseen fresh-online query requires four pre-routing host calls before its routed answer. FORGE-Lite is retrained with the 778 host-independent dimensions only: 768 BGE embedding dimensions, four BM25 statistics, one query–top-passage cosine, and five query-structure features. It uses no host probe in training or inference. Structured features are standardized using training-set statistics.

The strong BGE+BM25 baseline uses 773 host-independent dimensions (768 BGE, four BM25, and one query–top-passage cosine) and matches the Stage-0 data, MLP capacity, optimizer, and development-set search budget. It has no query-structure features, host probes, or GRPO. Table S7 reports the canonical cached feature-and-training ladder. Its point estimates should not be combined with the nine-run means in the uncertainty analysis as though they were the same statistic.

Table S5: Router architecture and three-stage training hyperparameters.

Hyperparameter	Value
Architecture	
Shared encoder $h(x)$	2-layer MLP, $789 \to 256 \to 256$, ReLU, dropout 0.1
Support-form head π^{μ}	Linear, 3 logits
Thinking-depth head π^{θ}	Linear, $\lvert\Theta\rvert$ logits, input $[h(x); e(\mu)]$ with $e(\mu) \in \mathbb{R}^{16}$
Total parameter count	$\sim$269K
Stage 0: Offline arm enumeration	
Warm-start alphabet $\mathcal{A}_{\text{warm}}$	$\mathcal{M} \times \{\text{NoThink}\}$ (3 arms)
Warm-start size N_{warm}	1,500 queries
Stage 1: Supervised warm-start (KL distillation)	
Soft-label temperature τ_0	1.0
Optimizer	AdamW
Learning rate	2×10^{-4}
Weight decay	0.01
Batch size	64
Maximum epochs	50
Early-stopping patience	7 (dev accuracy)
Stage 2: policy-guided GRPO refinement	
Full alphabet $\mathcal{A}$	K=6 (non-thinking hosts) or K=12 (thinking hosts)
Rollout group size G	8
Batch size B (queries per step)	32
Inner gradient steps per rollout	$K_{\text{inner}} = 4$
PPO clip range η	0.2
KL anchor target	0.02 (per query, adaptive β)
KL anchor β initial	0.05; multiplied by 1.5 every 100 steps if KL out of band
Entropy bonus α	0.01
Sampling	Stratified to ensure μ coverage when $G \geq \lvert\mathcal{M}\rvert$
Optimizer	AdamW
Learning rate	1×10^{-5}
Total update steps T_{grpo}	5,000
Pareto utility (default)	
Input cost weight λ_{in}	0.1 (swept in Table S14)
Output cost weight λ_{out}	0.2 (swept in Table S14)

Table S6: Feature groups for Full FORGE (789 dimensions) and FORGE-Lite (778 dimensions). Only Full uses the 11 host-dependent dimensions.

Group	Features	Dim
Query embedding (BGE)	Frozen sentence embedding	768
Query structural	length, WH flag, entity count, comparison flag, temporal flag	5
Retrieval	BM25 top-1 score, top-5 mean, score gap, score std	4
Host probe	answer length, output tokens, IDK flag, hedging flag, query overlap, numeric flag, confidence	7
Self-consistency	agreement, unique ratio, average length, greedy match	4
Cross-modal	query–top-1-passage cosine similarity	1
Full total	All six groups	**789**
Lite total	BGE + query structural + retrieval + cross-modal	**778**

B.6 Dataset Configuration

Table S8 lists the train/test splits per host group. The original tables use a canonical split (data-sampling seed 42), and cross-host transfer uses a matched canonical test subset. The uncertainty analysis combines three independent train/dev/test splits with three training seeds per split on both main local hosts. Its nine-run mean and standard deviation are distinct from canonical point estimates and paired-query bootstrap intervals on the canonical test split.

Table S7: **Canonical cached F1 (%) for matched feature and training variants.** The 773-dimensional BGE+BM25 baseline supplies a competitive reference. FORGE-Lite adds the five structural features without host calls; Full FORGE adds 11 host-dependent dimensions. All entries are point estimates on the canonical evaluation split, separate from the nine-run means.

Variant	Dimensions	Qwen3-8B	Mistral-7B
BGE+BM25-Hard	773	56.8	46.7
BGE+BM25-KL	773	57.5	47.5
FORGE-773 (three arms)	773	58.0	48.0
FORGE-773 (six arms)	773	58.4	48.6
FORGE-Lite	778	58.7	49.1
Full FORGE	789	59.5	50.1

Table S8: Per-benchmark sample sizes for the canonical split (data-sampling seed 42). The additional nine-run uncertainty analysis uses three independent splits and three training seeds per split.

Benchmark	**Train (local)**	**Test (local)**	**Test (transfer)**
HotpotQA (distractor)	1,500	300	75
2WikiMultiHopQA	1,500	300	75
MuSiQue	1,500	300	75
PopQA	1,500	300	75
FEVER	1,500	300	75

B.7 Asset Licenses and Attribution

All datasets, frozen language models, and software libraries used in this work are credited to their original creators, accessed under their original licenses, and used in a manner consistent with their intended research use. Table S9 summarizes the assets and their licenses; the underlying datasets and model weights are not redistributed with this work.

Table S9: Licenses for existing assets used in this work. Datasets are accessed via their original release channels; frozen LLMs are accessed via their published weights or provider APIs.

Asset	**Type**	**License / Terms**	**Citation**
HotpotQA (distractor)	Dataset	CC BY-SA 4.0	Yang et al. (2018)
2WikiMultiHopQA	Dataset	Apache 2.0	Ho et al. (2020)
MuSiQue	Dataset	CC BY 4.0	Trivedi et al. (2022)
PopQA	Dataset	MIT	—
FEVER	Dataset	CC BY-SA 3.0	—
Wikipedia (retrieval corpus)	Corpus	CC BY-SA 4.0	—
Qwen3-8B-Instruct / -Thinking	Frozen LLM	Apache 2.0	—
Qwen3.5-397B	Frozen LLM	Apache 2.0	—
Mistral-7B-Instruct-v0.3	Frozen LLM	Apache 2.0	—
Llama-3.3-70B-Instruct	Frozen LLM	Llama 3.3 Community	—
DeepSeek-V3.2 / -R1	Frozen LLM	MIT	—
Claude Sonnet 4	Frozen LLM (API)	Anthropic API terms	—
BGE-base-en-v1.5	Embedder	MIT	Xiao et al. (2024)
`rank_bm25`	Library	Apache 2.0	—
PyTorch / HuggingFace Transformers	Library	BSD / Apache 2.0	—

B.8 Compute Scale

Table S10 summarizes the overall compute footprint. At peak, **128 GPU accelerators ran concurrently** across 16 nodes for Stage 0 arm enumeration on the local backbones. Stage 2 GRPO refinement uses a single 8-GPU node per Pareto operating point. API-based experiments with frontier proprietary models were executed on externally managed infrastructure and incur no local cluster compute time. Stage 2 is a one-time source-host training cost, not a target-host cost incurred on each transfer query. In the matched six-action, nine-run comparison, it adds 1.8 macro-F1 points on

Qwen3-8B and 1.7 on Mistral-7B over the corresponding Stage-1 checkpoints, while reducing cached selected-answer tokens by 7.1% and 8.1%, respectively. The larger three-arm-to-six-arm difference also includes action-alphabet expansion and must not be attributed wholly to GRPO. Skipping Stage 2 avoids rollout cost but leaves Full FORGE's Fresh Online host probes.

Table S10: Compute resources and experiment counts.

Resource or artefact	**Value**
Cluster nodes used (unique)	16+
Peak concurrent GPU accelerators	**128**
SLURM allocations active	2 institutional allocations
Stage 0 host calls (per local backbone)	$\sim$4,500 (3 arms $\times$ 1,500 queries)
Stage 2 completions (per Pareto point)	1.28M ($T \cdot B \cdot G$ at T=5000, B=32, G=8)
Stage 2 input / output tokens (per point)	About 384M / 82M
Stage 2 wall-clock (Qwen3-8B local, per Pareto point)	8 to 12 hours on 8-GPU node
Stage 2 local GPU-hours (per point)	64 to 96 GPU-hours
API calls issued (transfer + thinking)	$\sim$22,000
Distinct LLM backbones evaluated	8
Parameter range	7B–671B
Benchmarks evaluated	5
Host $\times$ benchmark cells	40

The matched six-action budget trajectory separates GRPO refinement from action expansion. At 0, 2,500, and 5,000 updates, Qwen3-8B reaches 57.6, 58.9, and 59.4 macro-F1 with 0.255, 0.244, and 0.237k cached selected-answer tokens per query. Mistral-7B reaches 48.3, 49.5, and 50.0 macro-F1 with 0.272, 0.259, and 0.250k tokens. These improvements require the one-time training expenditure above; they do not account for Full FORGE's Fresh Online probes on later queries.

B.9 Inference Latency

Table S11 isolates local feature and router computation. The frozen BGE encoder dominates this local component, with little MLP time, but the table does not include the host executions required to populate Full FORGE's 11 host-dependent features on a fresh query. Main-text Table 2 reports end-to-end Fresh Online latency with those calls included.

Table S11: **Local routing latency (ms/query).** Median over 1,000 HotpotQA-style queries.

Component	**CPU** (1 thread)	**GPU** (batch 1)	**GPU** (batch 32, amort.)
BGE-base encoder (frozen)	27.4	4.9	0.39
Structured feature extraction	0.6	0.6	0.05
FORGE MLP head (π^{μ}, π^{θ})	1.1	0.05	0.01
Total routing decision	**29.1**	**5.5**	**0.45**
Self-consistency probe (K_{sc}=3 host samples)	—	—	90.0
Host answer inference (Qwen3-8B, 64 out tokens)	—	122.5	78.4

Mean query length is 18 tokens. The total routing-decision row covers local computation only. On a fresh Full FORGE query, three self-consistency samples and one greedy probe require host calls; the self-consistency row is an amortized batch-32 component, not complete Fresh Online overhead. Host answer inference is shown for scale.

In the cached-feature setting, a previously processed query can reuse the four Full FORGE probe outputs, leaving only the local routing computation before the selected answer. A new query in Fresh Online use must acquire those features: one greedy probe and three self-consistency samples precede the routed answer, for five host calls in total. FORGE-Lite retrains on host-independent features and needs only one host call for the routed answer.

Full FORGE trades much higher latency for its additional F1 on these local hosts: Qwen3-8B's p50 increases from 122.5 to 271.4 ms relative to Always-Raw. Lite obtains a smaller F1 gain with p50 near Always-Raw. On the measured DeepSeek-V3.2 API example, zero-shot Lite obtains 63.1 F1, 0.182k tokens, and about 2.2 s end-to-end latency versus Always-Raw's 63.4 F1, 0.287k tokens, and

about 2.1 s. This is a token-saving trade-off with slightly lower F1 and higher latency, not a three-axis Pareto improvement. Fresh Online Full FORGE latency was not measured on API hosts.

Cost-accounting note. Cached-result tables count selected-answer tokens and exclude earlier feature-acquisition calls, so they do not represent the end-to-end cost of a new query. Fresh Online evaluation also counts Full FORGE's greedy and self-consistency probes in tokens and latency; Lite has no host-dependent calls. Summary uses no auxiliary LLM call (Section B.3), and its prompt tokens are included in both cost-accounting protocols.

B.10 Reproducibility

Seed 42 identifies the canonical data split used for the point-estimate tables. The uncertainty analysis on Qwen3-8B and Mistral-7B instead evaluates three independent train/dev/test splits with three router-training seeds per split (nine runs per method and host). Each tunable baseline is retuned on its corresponding dev split with the same search budget. Nine-run standard deviations summarize split and initialization variation; paired-query bootstrap confidence intervals compare methods on the canonical test split and are not nine-run intervals. The bootstrap resampling unit is the matched test query. The *Oracle (6-arm)* reference rows are enumerated references that use the observed outcomes of all six arms for each query. They are not deployable policies and are reported as descriptive references rather than F1 or EM upper bounds. Exact reproduction requires model outputs, Stage-0 enumeration tables, Stage-1 warm-start checkpoints, and Stage-2 GRPO checkpoints in addition to the method settings reported here. Given pre-enumerated arm outputs, the reported Stage-1 supervised pipeline completes on one CPU in under two minutes. Stage-2 reproduction additionally requires the host endpoints used for rollout.

Hyperparameter selection. The Pareto utility's cost weights $(\lambda_{\text{in}}, \lambda_{\text{out}}) = (0.10, 0.20)$ and the Boltzmann temperature $\tau = 1.0$ are selected once on the Qwen3-8B HotpotQA dev set (N=500) and held fixed across the original host and benchmark comparisons. These weights specify a study operating point, not a universal price for user value. Table S14 reports a canonical cached sensitivity grid; its F1 range is 1.2 points on the non-thinking host and 2.5 points on the thinking-capable host. Baselines with tunable scalars receive the same dev-budget tuning protocol on the same dev split: the BM25-Threshold rule's two thresholds are grid-searched, Adaptive-RAG's complexity-classifier threshold is calibrated, and the Sysformer adapter rank is selected from $\{4, 8, 16\}$. In cross-host transfer (Section 4.6), λ and τ are not retuned per target host and no target-host Stage 2 training is run. Zero-shot describes the transferred weights, not the absence of Full FORGE's target-host feature probes on a fresh query; the API transfer tables use cached-feature accounting.

B.11 Supplementary Figures

This subsection collects the schematic of the three extrinsic axes of adaptive inference (Figure S2).

C Additional Ablations

This appendix collects ablations referenced from the main text but deferred for space.

C.1 Stage-2 Training and KL-Anchor Ablations

Table S12 puts the Stage-2 algorithm comparison and KL-anchor sweep on the same two benchmarks and cost scale. The shared three-arm Stage-1 row is a pipeline reference rather than a same-alphabet GRPO control; Panel A compares six-arm refinements, and Panel B varies the GRPO anchor around the default row in Panel A. The matched six-arm, nine-run comparison reported above isolates Stage 2 from alphabet expansion. GRPO and Dr. GRPO differ by at most 0.2 F1 in the displayed point estimates on the two benchmarks. Both exceed DPO and RLOO here, but this table alone does not establish a statistical tie or measure the matched six-action Stage-2 gain across all five tasks. That controlled comparison gives $+1.8$ and $+1.7$ macro-F1 on Qwen3-8B and Mistral-7B, respectively, at 1.28M model completions per operating point (Appendix B.8). GRPO is the reported primary configuration; the comparisons among DPO, RLOO, Dr. GRPO, and GRPO apply to this implementation and tested action alphabets.

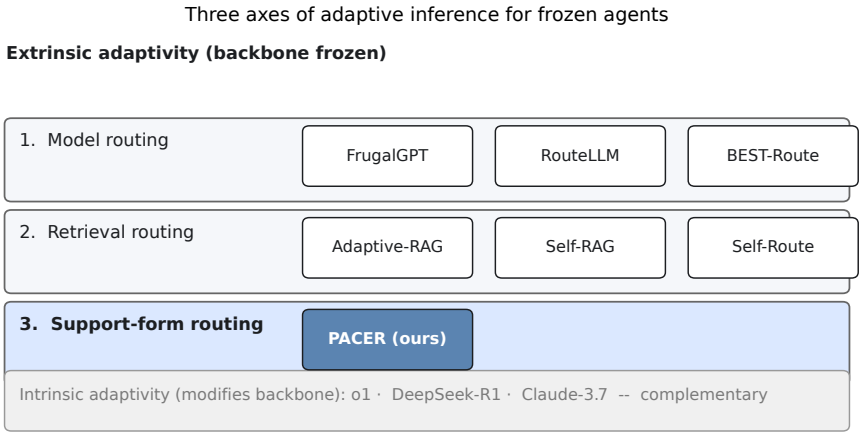

Figure S2: **Three extrinsic axes of adaptive inference for frozen agents.** Model routing (Chen et al., 2024; Ding et al., 2025) and retrieval routing (Jeong et al., 2024; Asai et al., 2024) have been studied at length. Support-form routing, the third axis, is the focus of this paper. An orthogonal intrinsic axis exists on hosts that expose a reasoning budget (OpenAI et al., 2024; Guo et al., 2025), and our framework composes with it where available.

Table S12: **Stage-2 refinement and KL-anchor ablations on Qwen3-8B.** Canonical Cached F1 (%) and selected-answer token cost $\bar{C}$ (k) on HotpotQA and MuSiQue. Panel A compares six-arm refinement methods from the same Stage-1 checkpoint. Panel B varies β for six-arm GRPO; the highlighted GRPO row in A is its default adaptive β=0.05 point. The three-arm Stage-1 row is shown once as a shared pipeline reference, not a matched six-arm β=∞ limit.

Configuration	HotpotQA		MuSiQue		Avg F1 ↑
	F1 ↑	$\bar{C}$ ↓	F1 ↑	$\bar{C}$ ↓	
Shared three-arm pipeline reference					
Offline KL only (FORGE (Stage 1))	58.3	0.272	20.8	0.295	39.6
A. Stage-2 algorithm (six-arm alphabet)					
DPO (Rafailov et al., 2023)	59.5	0.258	25.0	0.265	42.3
RLOO (Ahmadian et al., 2024)	59.2	0.263	24.6	0.272	41.9
Dr. GRPO (Liu et al., 2025)	60.6	0.235	26.5	0.249	43.6
GRPO (β=0.05, default)	**60.4**	**0.236**	**26.4**	**0.252**	**43.4**
B. GRPO KL-anchor coefficient β (six-arm alphabet)					
$\beta = 0$ (no anchor)	56.2	0.412	22.1	0.398	39.2
$\beta = 0.01$ (weak)	59.6	0.252	25.5	0.265	42.6
$\beta = 0.20$ (strong)	59.1	0.265	24.7	0.282	41.9

C.2 Factorized versus Flat Policy Head

Table S13 compares the default factorized head $\pi_\theta(a \mid x) = \pi_\theta^\mu(\mu \mid h(x)) \cdot \pi_\theta^\theta(\theta \mid h(x), \mu)$ against a flat K-way categorical head over the joint action $a = (\mu, \theta)$, with feature vector and Stage-2 hyperparameters held fixed. At K=6 the displayed factorized-versus-flat difference is 0.6 F1; at K=12 it is 1.6 F1, with the reported parameter counts 269K versus 273K. These two tested settings are consistent with a benefit from conditioning thinking depth on support form, but do not establish a general scaling law in K. All main-paper results use the factorized head.

Table S14 shows a quality–cost trade-off rather than invariance to the cost weights: the non-thinking host spans 1.2 F1 points and the thinking-capable host spans 2.5 points across the displayed grid. The table uses cached selected-answer tokens; the Fresh Online token curves also include feature acquisition. Increasing λ_{in} shifts the policy toward cheaper input arms (Direct over Raw), and increasing λ_{out} specifically suppresses thinking-budget arms on thinking-capable hosts. The default of $(0.1, 0.2)$ was selected on a held-out development set and held fixed for all main results.

Table S13: **Factorized vs flat policy head.** FORGE F1 (%), average cost $\bar{C}$ (k), and policy parameter count under two architectures: a flat K-way categorical head over the joint action $a = (\mu, \theta)$, and the default factorized head $\pi^{\mu}(\mu \mid h(x)) \cdot \pi^{\theta}(\theta \mid h(x), \mu)$ (Section 3.1). Reported on `Qwen3-8B` (K=6) and `Qwen3-8B-Thinking` (K=12).

Policy head	`Qwen3-8B` (K=6)		`Qwen3-8B-Thinking` (K=12)		**# params**
	F1↑	$\bar{C}\downarrow$	F1↑	$\bar{C}\downarrow$	
Flat K-way categorical	59.8	0.241	63.2	0.715	273K
Factorized $\pi^{\mu} \cdot \pi^{\theta\|\mu}$ (default)	**60.4**	**0.236**	**64.8**	**0.690**	**269K**

Table S14: **Canonical cached sensitivity to cost weights λ_{in} and λ_{out}.** FORGE F1 (%) and selected-answer token cost $\bar{C}$ (k) on `HotpotQA` for non-thinking Qwen3-8B and thinking-capable Qwen3-8B-Thinking. Across the surveyed points, F1 spans 1.2 and 2.5 points, respectively. The study default $(0.1, 0.2)$ is highlighted; these weights are not universal prices.

λ_{in}	λ_{out}	Non-thinking host		Thinking-capable host	
		F1↑	$\bar{C}\downarrow$	F1↑	$\bar{C}\downarrow$
0.05	0.0	60.8	0.262	65.5	0.852
0.05	0.2	60.7	0.246	65.0	0.731
0.10	0.0	60.4	0.253	65.2	0.802
0.10	0.20 (default)	**60.4**	**0.236**	**64.8**	**0.690**
0.10	0.50	59.9	0.218	63.5	0.582
0.20	0.20	59.6	0.210	63.0	0.624

Table S15: **Training-size sweep.** HotpotQA F1 (%) for Stage 1 and Stage 1+2 on Qwen3-8B.

N_{warm}	FORGE (Stage 1)	FORGE (Stage 1+2)
200	46.8	50.6
500	**58.5**	**60.2**
1,000	58.2	60.4
1,500 (default)	**58.3**	**60.4**
2,000	58.4	60.5

Table S15 shows empirical Stage-1 saturation around $N_{\text{warm}} = 500$ on this HotpotQA sweep; no finite-sample theorem here predicts that threshold. The Stage 1-to-Stage 1+2 difference ranges from $+1.7$ to $+3.8$ F1 across the displayed training sizes, and is $+2.1$ at the default $N_{\text{warm}} = 1{,}500$. This single-benchmark sweep does not isolate GRPO from action-alphabet changes in the main three-arm-to-six-arm comparison. Stage 2 needs rollout calls but no extra warm-start data.

C.3 Per-Arm Allocation Changes from Three to Six Actions

The original three-arm Stage-1 to six-arm final-policy comparison differs in both action availability and optimization; its $+3.5$ Qwen3-8B and $+3.6$ Mistral-7B macro-F1 differences are not GRPO-only effects. The matched six-action, nine-run comparison isolates the Stage-2 increment at $+1.8$ [1.3, 2.3] and $+1.7$ [1.2, 2.2] macro-F1, respectively (paired 95% confidence intervals on the canonical test split). Table S16 describes the canonical three-arm-to-six-arm allocation changes on Qwen3-8B. Its per-arm terms are descriptive accounting terms, computed as $\Delta\text{alloc}_a \cdot (m_a^{\text{F1}} - m_{\text{ref}}^{\text{F1}})$ for the selected query subsets; they should not be interpreted as controlled causal effects of GRPO. Here CoT-Prompt aggregates the three CoT-Prompt arms across support forms.

The largest descriptive term is the $+2.9$ associated with CoT-Prompt, which the three-arm warm-start policy cannot select. Direct and Summary shifts contribute $+1.3$ combined, while the Raw term is -0.7 in this accounting. These terms describe the changed allocations and query subsets; they do not partition the effect of GRPO from that of adding CoT-Prompt. The matched six-arm comparison above provides that controlled Stage-2 estimate.

Table S16: **Descriptive per-arm accounting for Qwen3-8B's three-arm Stage-1 to six-arm final-policy comparison.** Allocations sum to 100% within each stage, and the displayed terms sum (up to rounding) to the observed $+3.5$ macro-F1 difference. The action alphabet also changes, so this is not the controlled GRPO-only estimate; Mistral-7B's aggregate difference is $+3.6$.

Arm a	Stage-1 alloc. (%)	Final alloc. (%)	Δ alloc. (pp)	Descriptive F1 term
Direct	13	18	$+5$	$+0.4$
Summary	27	32	$+5$	$+0.9$
Raw	60	35	-25	-0.7
CoT-Prompt	0	15	$+15$	$+2.9$
Total	100	100	—	**+3.5**

C.4 Router Behavior in BGE Embedding Space

This subsection describes routing in BGE embedding space (Figure S3), and against the per-instance utility oracle (Figure S4). These plots show action allocation without identifying the cause of each answer. In these figures, CoT-Prompt aggregates the CoT-Prompt arms across support forms, and Direct, Summary, and Raw denote the corresponding NoThink arms.

Per-point structure in BGE embedding space. Figure S3 plots a 2D UMAP projection of the BGE-base sentence embeddings of the same $N{=}1{,}500$ held-out test queries, colouring each point by FORGE's argmax action. Five visible cluster regions correspond to five query types derived from dataset-provided labels (*Comparison*, *Factoid*, *Multi-hop 2–3 hops*, *Multi-hop 4+ hops*, *Verification*); they emerge from the BGE embedding alone, before the router sees them. Most clusters are dominated by a single arm, and mixed colours concentrate near cluster boundaries. The regions are consistent with query-dependent routing in the displayed BGE projection, but the visualization alone cannot rule out memorization or establish why an individual action succeeds.

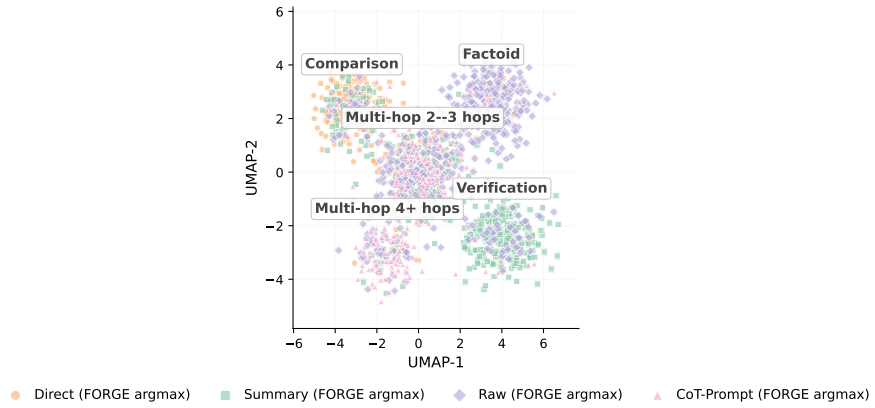

Figure S3: **2D UMAP projection of held-out BGE query embeddings ($N{=}1{,}500$), coloured by FORGE's argmax action on Qwen3-8B.** The five visible clusters correspond to five query types derived from dataset labels (annotated near each cluster centroid). UMAP uses frozen BGE-base query embeddings, $n_{\text{neighbors}}{=}30$, $\text{min\_dist}{=}0.3$, and cosine distance; centroid labels are illustrative.

Alignment with per-instance oracle. Figure S3 shows that FORGE's routing is structured in embedding space but not yet that the structure is *correct*. Figure S4 places FORGE's routing beside the per-instance *utility oracle* on the same UMAP projection. This diagnostic oracle evaluates all six actions and selects the one maximizing the stated F1-minus-token-cost utility; it is separate from the three-arm Stage-0 warm-start enumeration and from the tabular Oracle reference rows. FORGE selects its answer action without enumerating all arms, although Full FORGE still needs host-dependent feature probes on a fresh query. The two panels are nearly indistinguishable: **FORGE matches the per-instance oracle on 87% of queries**, with the residual 13% disagreement concentrated at the boundaries between clusters where multiple arms are near-tied in Pareto utility (visible as the small fraction of mismatched-colour points within each cluster, especially on the multi-hop 2–3-hop

region where Raw and CoT-Prompt are near-tied for many queries). The 87% agreement is with the specified utility oracle; the UMAP plot does not explain the outcome of individual answers.

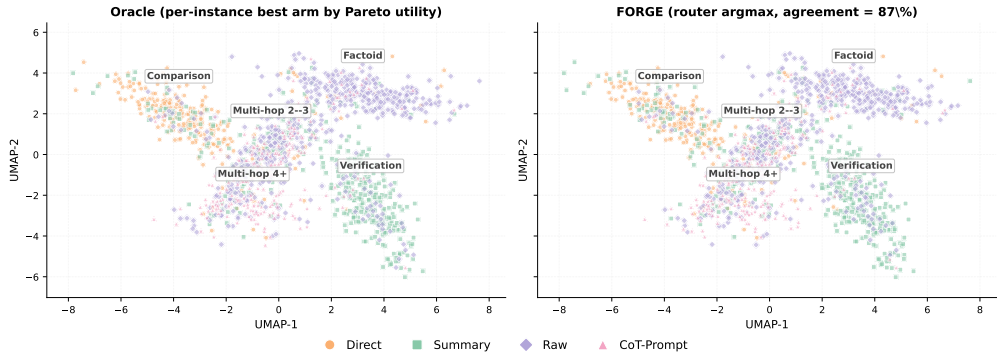

Figure S4: **FORGE's argmax action agrees with the six-arm utility oracle on 87% of the same queries (N=1,500, Qwen3-8B).** *Left:* the arm maximizing the stated F1-minus-token-cost utility after offline six-arm evaluation; this diagnostic is separate from the tabular Oracle reference rows and from three-arm Stage 0. *Right:* the trained router's argmax without answer-arm enumeration. Coordinates and cluster centroids are shared. The residual 13% action disagreement is concentrated near the displayed multi-hop cluster boundary; this plot does not measure Fresh Online probe cost.

C.5 Cross-Host Transfer: Per-Task Breakdown

Table 8 reports macro F1 for cross-host transfer; Table S17 adds per-task F1, Always-Direct (the parametric-only reference), and FORGE Stage 1 (the supervised warm-start ablation).

C.6 Exact Match (EM) Scores

The main paper reports token-level F1 throughout for direct comparability with prior routing work. Table S18 reports EM on the same canonical cached predictions as Table 1. Its *Oracle (6-arm)* row is the enumerated reference of Table 1 scored under EM (Appendix B.10); it is a descriptive reference, not an EM upper bound. The relative ordering is largely preserved, with FORGE leading the non-oracle policies in macro EM on both hosts. EM/F1 ratios are roughly 0.78 on HotpotQA, 0.82 on 2Wiki, 0.65 on MuSiQue (multi-hop tolerates partial overlap), 0.87 on PopQA, and 0.96 on FEVER (three-way classification).

Table S17: **Zero-shot cross-host transfer: per-task F1 and cached cost.** A source-trained FORGE router is applied to three API hosts without target-host weight updates.

Target host	Policy	F1 by task (%)					**Avg F1** ↑	$\bar{C}$ (k) ↓ saving	$\Delta^{F_1}_{\text{Raw}}$
		HQA	2Wiki	MuSiQue	PopQA	FEVER			
`Llama-3.3 70B`	Direct	42.9	32.4	13.4	38.5	61.3	37.7	0.085 (−73%)	-20.8
	Summary	57.6	37.1	20.5	90.3	74.7	56.0	0.214 (−33%)	-2.5
	Raw	**64.7**	41.5	26.9	88.8	70.7	58.5	0.318 (ref.)	+0.0
	Oracle	*73.8*	*54.7*	*36.3*	*90.8*	*76.0*	*66.3*	*0.145* (−54%)	—
	Stage 1	61.9	45.3	24.1	90.3	72.0	58.7 ± 0.4	0.215 (−32%)	+0.2
	FORGE	64.0	**47.8**	**27.2**	**91.5**	**75.5**	**61.2 ± 0.6**	0.205 (−36%)	+2.7
`DeepSeek V3.2`	Direct	48.2	29.2	22.5	39.6	49.3	37.8	0.055 (−81%)	-25.6
	Summary	58.4	43.3	28.3	**94.8**	**88.0**	62.6	0.184 (−36%)	-0.8
	Raw	**65.6**	42.8	31.8	91.3	85.3	**63.4**	0.287 (ref.)	+0.0
	Oracle	*72.2*	*61.0*	*44.4*	*96.1*	*96.0*	*73.9*	*0.116* (−60%)	—
	Stage 1	63.3	42.8	30.7	92.9	72.0	60.3 ± 0.4	0.183 (−36%)	-3.1
	FORGE	65.4	**45.5**	**33.5**	94.0	78.5	**63.4 ± 0.7**	0.176 (−39%)	+0.0
`Qwen3.5 397B`	Direct	54.0	25.0	19.1	38.9	66.7	40.7	0.062 (−80%)	-22.4
	Summary	58.7	44.8	23.4	**95.6**	**88.0**	62.1	0.196 (−36%)	-1.0
	Raw	65.0	41.0	29.8	93.0	86.7	63.1	0.305 (ref.)	+0.0
	Oracle	*74.2*	*56.0*	*40.2*	*95.6*	*93.3*	*71.9*	*0.123* (−60%)	—
	Stage 1	63.1	46.5	30.7	94.5	82.7	63.5 ± 0.4	0.209 (−31%)	+0.4
	FORGE	**65.5**	**49.0**	**33.0**	94.5	86.0	**65.6 ± 0.5**	0.198 (−35%)	+2.5

HQA abbreviates HotpotQA. Direct, Summary, and Raw are fixed policies. Each task has 75 canonical test queries. $\bar{C}$ counts selected-answer tokens (k/query); its second line is the reduction relative to Raw. $\Delta^{F_1}_{\text{Raw}}$ compares macro F1 with Raw. Stage 1 and FORGE macro F1 are mean ± SD over three seeds on the fixed subset; per-task F1 and baseline scores are point estimates. The italic Oracle is an enumerated six-arm reference (Appendix B.10) and is excluded from practical-policy rankings. Bold marks the highest practical per-task score; boxed macro F1 marks FORGE, which ties Raw on DeepSeek-V3.2 at displayed precision. The router uses 2,000 source training and 500 development queries. Pre-routing host calls are excluded from cached costs.

Table S18: **Exact Match on the canonical cached predictions of Table 1.** Scores are percentages.

Host	Policy	EM by task (%)					**Avg EM ↑**
		HQA	2Wiki	MuSiQue	PopQA	FEVER	
Qwen3-8B	Direct	18.5	21.2	6.1	12.8	52.0	22.1
	Summary	36.5	31.0	12.2	75.5	**78.0**	46.6
	Raw	45.0	33.5	15.7	73.0	76.0	48.6
	Oracle (6-arm)	*53.5*	*43.8*	*22.6*	*76.5*	*86.0*	*56.5*
	Adaptive-RAG	44.0	33.0	15.0	69.5	74.5	47.2
	TierMem-2arm	37.0	32.5	13.5	71.0	**78.0**	46.4
	s3	43.0	32.0	14.0	68.0	72.0	45.8
	Sysformer	45.0	32.0	13.7	75.0	73.0	47.7
	Stage 1	45.5	31.5	13.5	76.0	72.5	47.8
	FORGE	**47.5**	**35.0**	**17.2**	**76.5**	77.0	**50.6**
Mistral-7B	Direct	17.5	14.3	4.7	19.5	45.0	20.2
	Summary	29.5	19.8	7.0	69.5	66.5	38.5
	Raw	36.5	23.5	8.4	65.0	69.0	40.5
	Oracle (6-arm)	*47.0*	*35.0*	*14.0*	*72.5*	*78.5*	*49.4*
	Adaptive-RAG	35.5	22.0	9.0	63.5	67.0	39.4
	TierMem-2arm	31.5	23.0	7.7	61.0	67.0	38.0
	s3	35.5	21.5	8.3	61.5	64.0	38.2
	Sysformer	37.0	19.0	8.5	65.0	66.0	39.1
	Stage 1	37.5	19.0	8.7	66.5	67.0	39.7
	FORGE	**39.5**	**25.0**	**10.0**	**70.0**	**69.5**	**42.8**

HQA abbreviates HotpotQA. Direct, Summary, and Raw are fixed policies. Adaptive-RAG (Jeong et al., 2024), TierMem-2arm (Zhu et al., 2026), s3 (Jiang et al., 2025b), and Sysformer (Sharma et al., 2025) are the comparison routers. The italic Oracle is the enumerated six-arm reference (Appendix B.10) and is excluded from the practical-policy ranking. Bold marks the best practical per-task EM (including ties); boxed macro EM marks the best practical policy on each host.

C.7 Numerical Table for the Action Alphabet Sweep

The KL-anchor values appear alongside the Stage-2 algorithm comparison in Table S12, Panel B, with the default point in Panel A. Table S19 reports the action alphabet sweep. Adding CoT-Prompt ($K{=}3 \to 6$) raises F1 while lowering cost, adding Think-Low ($K{=}6 \to 9$) gives modest gains ($+1.0$ and $+1.1$ F1), and unlocking Think-High ($K{=}9 \to 12$) gives the largest absolute improvement.

Table S19: **Action alphabet size ablation.** FORGE F1 (%) and cached selected-answer tokens $\bar{C}$ (k) on `HotpotQA` and `MuSiQue` as the alphabet grows from $K{=}3$ to $K{=}12$ on `Qwen3-8B-Thinking`.

K	Alphabet	HotpotQA		MuSiQue	
		F1 ↑	$\bar{C}$ ↓	F1 ↑	$\bar{C}$ ↓
3	{Direct, Summary, Raw} × {NoThink}	58.5	0.272	21.3	0.295
6	$\mathcal{A}_{\text{nt}}$ (default non-thinking)	60.4	0.236	26.4	0.252
9	$\mathcal{A}_{\text{nt}} \cup$ Think-Low arms	61.4	0.380	27.5	0.420
12	$\mathcal{A}_{\text{t}}$ (full 12-arm)	**64.8**	0.690	**33.5**	0.760

C.8 Heuristic Routing Baselines

Two heuristic routing baselines were excluded from the main results in Table 1 for compactness. **Self-Routing** uses the host's own confidence (a normalized log-probability of the Direct response) thresholded at the dev-tuned value to decide between Direct and Raw. **FrugalGPT-Cascade** (Chen et al., 2024) runs a Direct → Summary → Raw threshold cascade, advancing whenever the previous arm's confidence falls below a dev-tuned threshold. Both favor inexpensive actions in the reported

runs, but they do not have identical behavior: Self-Routing improves Mistral-7B macro F1 from 24.2 to 28.2 in Table S20. Its initial Direct confidence call, and the earlier calls in the cascade, must be included in any Fresh Online cost or latency comparison.

Table S20: **Heuristic routing baselines on the main hosts.** F1 (%) and cached selected-answer token cost $\bar{C}$ (k) on the five benchmarks of Table 1. Cached $\bar{C}$ omits preliminary Direct/confidence and cascade calls; thus this is not a Fresh Online comparison. Always-Direct is a reference row.

Host	Method	HotpotQA	2Wiki	MuSiQue	PopQA	FEVER	Avg F1	$\bar{C}$	$\Delta^{F_1}_{\text{Raw}}$
Qwen3-8B	Always-Direct	26.2	25.7	10.2	15.0	54.4	26.3	0.040	-30.8
	Self-Routing	27.0	27.6	11.1	18.2	56.1	28.0	0.058	-29.1
	FrugalGPT-Cascade	26.2	25.7	10.2	16.5	55.0	26.7	0.042	-30.4
Mistral-7B	Always-Direct	25.2	17.7	7.9	23.0	47.0	24.2	0.041	-23.0
	Self-Routing	32.4	24.2	11.1	24.0	49.5	28.2	0.060	-19.0
	FrugalGPT-Cascade	25.3	17.9	8.0	23.5	47.5	24.4	0.043	-22.8

D Properties of the Routing Objective and Warm Start

These results characterize the finite-action offline target and show how the warm-start reference shapes it; they give no convergence or regret guarantee for the implemented optimizer.

D.1 Value of Per-Query Routing

Proposition 1 (Value of per-query routing)**.** *For integrable utilities, define $U^*(x) = \max_{a\in\mathcal{A}} U(x,a)$ as the best achievable per-query utility. Then*

$$\mathbb{E}_x[U^*(x)] \;\geq\; \max_{a\in\mathcal{A}} \mathbb{E}_x[U(x,a)].$$

It is strict if every action a is suboptimal on a set of queries with positive probability.

Proof. For each a and x, $U^*(x) - U(x,a) \geq 0$. Taking expectations gives the weak inequality. Under the stated positive-probability condition, each nonnegative difference has strictly positive expectation. Since $\mathcal{A}$ is finite, the inequality is strict against the best fixed action. □

D.2 A Fixed-Reference Regularized Objective

The Stage 2 loss includes both an entropy bonus and a KL penalty to the warm-start policy. Their effect can be characterized for an idealized, unrestricted categorical policy with fixed coefficients.

Proposition 2 (Idealized regularized policy)**.** *Fix x, a full-support reference policy $\pi_0(\cdot \mid x)$, and coefficients $\alpha, \beta \geq 0$ with $\alpha + \beta > 0$. The unique maximizer over $\pi(\cdot \mid x) \in \Delta(\mathcal{A})$ of*

$$\sum_a \pi(a \mid x)U(x,a) - \beta D_{\mathrm{KL}}\big(\pi(\cdot \mid x) \,\|\, \pi_0(\cdot \mid x)\big) + \alpha H\big(\pi(\cdot \mid x)\big)$$

is

$$\pi^\dagger(a \mid x) = \frac{\pi_0(a \mid x)^{\beta/(\alpha+\beta)} \exp\big(U(x,a)/(\alpha+\beta)\big)}{\sum_{a'} \pi_0(a' \mid x)^{\beta/(\alpha+\beta)} \exp\big(U(x,a')/(\alpha+\beta)\big)}. \tag{9}$$

Proof. Expanding the KL divergence and entropy shows that the objective equals $(\alpha+\beta)\log Z(x) - (\alpha+\beta)D_{\mathrm{KL}}(\pi(\cdot \mid x) \,\|\, \pi^\dagger(\cdot \mid x))$, where $Z(x)$ is the denominator in Eq. (9). The KL divergence is nonnegative and vanishes only when the two policies coincide, proving a unique optimum. □

When $\beta = 0$, Eq. (9) reduces to Eq. (2) with $\tau = \alpha$. A nonuniform reference generally changes the solution when $\beta > 0$. The proposition describes the exact objective with fixed coefficients and known utilities. Stage 2 instead uses sampled, group-standardized rewards, a clipped policy surrogate, a finite router, and an adaptive KL coefficient; the proposition gives no convergence or regret guarantee for the implemented Stage 2 optimization procedure.

D.3 Warm-Start KL Decomposition

Stage 1 fits the support-form marginal on the enumerated warm-start actions. For the full action set, the thinking-depth head assigns equal initial probability to every thinking option.

Proposition 3 (Warm-start KL decomposition)**.** *Fix x. Let π^* be the full-action distribution of Eq.* (2)*, with support-form marginal π^*_μ and conditional thinking policy $\pi^*_{\theta|\mu}$. Suppose the warm-start policy factorizes as $\pi_{\theta_0}(\mu, \theta \mid x) = \pi^\mu_{\theta_0}(\mu \mid x)/|\Theta|$, with $\pi^\mu_{\theta_0}(\mu \mid x) > 0$ for all μ. Then*

$$\begin{aligned} D_{\mathrm{KL}}\big(\pi^*(\cdot \mid x) \,\|\, \pi_{\theta_0}(\cdot \mid x)\big) = D_{\mathrm{KL}}\big(\pi^*_\mu(\cdot \mid x) \,\|\, \pi^\mu_{\theta_0}(\cdot \mid x)\big) \\ + \log|\Theta| - \mathbb{E}_{\mu\sim\pi^*_\mu(\cdot|x)}\big[H(\pi^*_{\theta|\mu}(\cdot \mid x, \mu))\big]. \end{aligned} \tag{10}$$

The contribution from the conditional thinking policy lies in $[0, \log|\Theta|]$.

Proof. The KL chain rule splits the joint divergence into a divergence between support-form marginals and the π^*_μ-weighted conditional divergence. For each μ, $D_{\mathrm{KL}}(\pi^*_{\theta|\mu} \,\|\, \mathrm{Uniform}(\Theta)) = \log|\Theta| - H(\pi^*_{\theta|\mu})$. Substitution gives Eq. (10); the interval follows from $0 \le H(\pi^*_{\theta|\mu}) \le \log|\Theta|$. □

This identity separates support-form marginal mismatch from mismatch due to a uniform thinking head. It bounds only the latter; the marginal term can be larger. The identity describes initialization mismatch and does not predict the empirical F1 improvement from Stage 2.

E Interpretation of the Cost–Quality Score

Equation (1) is a weighted-sum scalarization of measured F1 and normalized token costs. Its weights select an operating point according to the user's relative valuation of these quantities; performance away from the reported weight sweep is an empirical question. Without the entropy term, maximizing expected utility over all categorical policies chooses a utility-maximizing action for each query (with arbitrary mixing among ties). The Shannon-entropy regularizer yields the soft labels in Eq. (2).

The measured token count in this formulation is an operational resource cost. It is not the mutual-information rate in Shannon's rate–distortion function, so the cost-quality scalarization alone does not constitute a rate–distortion derivation of the Boltzmann policy.

F Limitations and Future Work

F.1 Limitations

The scope of this paper is knowledge-intensive QA and fact verification, the task families on which the Direct/Summary/Raw alphabet is most directly meaningful; reasoning-only benchmarks (e.g., MATH, GSM8K, AIME) where retrieval is rarely useful would require a different action alphabet (e.g., chain-of-thought depth or tool calls) and are not evaluated here. Stage 0 enumeration cost remains linear in $|\mathcal{A}_{\mathrm{warm}}|$, which constrains how rich the warm-start alphabet can be at training time; Stage 2 GRPO removes this constraint for the full alphabet $\mathcal{A}$ but introduces a host-call rollout budget of roughly $T_{\mathrm{grpo}} \cdot B \cdot G$ per Pareto operating point. FORGE is trained against a fixed retrieval backend, so joint optimization of retrieval and routing is beyond the scope of this work. In addition, the current router relies on compact query-level and retrieval-derived features as input; for document-understanding tasks where long-range context, layout structure, or cross-document interactions are central, and especially for multimodal settings involving image-text evidence, a context-aware or multimodal router may be needed to fully capture the support requirements. On tasks where a single support form uniformly dominates, notably FEVER on non-thinking hosts where Summary $\approx$ Raw for 96% of queries, the general-purpose router does not improve on a task-specific fixed policy; on thinking-capable hosts this gap closes once the thinking-arm composition restores routing dynamic range. Transfer accuracy is sensitive to the capability gap between source and target hosts: a source-trained router's support-form allocation can mismatch the target host's parametric knowledge, as observed on FEVER with DeepSeek-V3.2, where FORGE trails both fixed Summary and fixed Raw. Stage 0 arm enumeration and Stage 2 training require labeled queries to compute the F1 reward, although inference with a trained router does not require labels. An LLM judge or verifier could supply proxy rewards for unlabeled adaptation, pending validation against true answer quality.

F.2 Future Work

Three directions follow from the current formulation. A target-side calibration step that updates the last linear layer from a small labelled subset of the target host is a natural remedy for the source-target capability-gap limitation above. Coupling support-form routing with retrieval or model routing would enlarge the discrete action alphabet; its training cost and empirical value remain to be tested. Extending Stage 2 to a sequential cascade, in which the router decides whether to escalate from Direct to Summary to Raw based on intermediate host responses, would require accounting for the extra host calls and evaluating the resulting policy separately. Replacing BM25 with dense (DPR or Contriever) or hybrid sparse-dense retrieval changes the upstream retriever, while $\mathcal{M}$ still describes what enters the host prompt. Comparing these backends could test the scope of FORGE's improvements in answer quality and token efficiency.

F.3 Broader Impacts

FORGE changes which evidence and reasoning setting a frozen host receives, so its quality and cost effects depend on the task, host, and feature-acquisition protocol. The cached-feature comparisons measure routed-answer tokens after Full FORGE's host-dependent features have been collected. In the measured Fresh Online Qwen3-8B setting, Full FORGE improves F1 from 57.1 to 59.5 and reduces host tokens from 0.430k to 0.384k per query relative to Always-Raw, but its four pre-routing host calls raise median latency from 122.5 to 271.4 ms. FORGE-Lite avoids those calls and reaches 58.7 F1 at 0.242k tokens and 128.0 ms. These measurements show a deployment trade-off; token counts alone do not establish energy savings or lower end-to-end cost for every provider.

The cost weights $(\lambda_{\text{in}}, \lambda_{\text{out}})$ encode a chosen quality–cost preference. More aggressive cost weighting can select less evidence and may worsen answers or remove the grounding needed for attribution. A strong fixed form can also be preferable on a particular task: on FEVER with a non-thinking host, the Summary-pinned baseline slightly exceeds the general router's F1. In settings where citations or other grounding requirements matter, they should be measured directly and included as deployment criteria rather than assumed to follow from F1 or a change in cost weights. Although the host weights remain frozen, changing its inputs can change its errors; we have not evaluated whether routing amplifies or reduces bias, hallucination, or misuse risks outside the reported tasks.

G Use of Large Language Models

An AI assistant helped revise the manuscript's prose, organization, and internal consistency, including reviewing theoretical claims for assumptions and scope. The authors are responsible for verifying the technical claims, references, and empirical results in the submitted version.