RouterWorld: Scaling Conversational AI with Latency-Aware LLM Routing

Anonymous ACL submission

Abstract

Conversational AI systems built on Large Language Models (LLMs) require routing mechanisms that balance answer quality and response latency at scale. However, existing LLM routing benchmarks assume full observability of all candidate model outcomes, under-emphasize latency as a first-class metric, and fragment evaluation across heterogeneous protocols misaligned with real-world settings. We introduce **RouterWorld**, a comprehensive benchmark for LLM routing in conversational GenAI. RouterWorld unifies **371k** single-turn responses and **76k** multi-turn agentic trajectories, adopts natural user interactions with a standardized LLM-as-a-judge protocol, and decouples efficiency from transient pricing via hardware-grounded latency annotations. Crucially, RouterWorld formalizes routing under partial observability through two setups: **Cold-Start**, which restricts learning to sparse, static logs, and **Voyager**, which models continuous data streams where routing decisions shape future training distributions. Evaluating 13 routing baselines, we find that most routers over-emphasize accuracy while ignoring latency under partial observability. All code and data are publicly available.

1 Introduction

Conversational AI systems powered by large language models (LLMs) increasingly operate at massive scale, serving millions of user queries per minute across chat assistants (Team et al., 2023; Llama, 2023; Chiang et al., 2024; Amazon, 2025), search interfaces (Chen et al., 2025; Liu et al., 2023), and interactive agents (Yao et al., 2023; Wu et al., 2024; Qin et al., 2024). In these settings, every query is user-facing and latency-sensitive. Rather than relying on a single monolithic model, modern conversational systems deploy heterogeneous pools of LLMs that vary widely in capability, specialization, and inference latency. Intelligent routing, which dynamically selects the most appropriate model for each query, has thus emerged as a central mechanism for scaling Generative AI while balancing response quality and cost.

Recent work has explored various routing strategies, including supervised classifiers trained on offline logs (Ong et al., 2025; Ding et al., 2024; Feng et al., 2025b), bandit-based methods that optimize routing policies via online interaction (Panda et al., 2025; Wei et al., 2025), and cascade architectures that progressively escalate queries to stronger models (Chen et al., 2024). To facilitate comparison across methods, benchmarks have been proposed (Hu et al., 2024; Li et al., 2026; Huang et al., 2025).

Despite this progress, existing evaluation methodologies for LLM routing remain misaligned with real-world deployment constraints. A central limitation is the widespread assumption of full observability: as illustrated in Figure 1 (Left), standard protocols assume the router has access to correctness and efficiency outcomes of *all* candidate models per query. This diverges sharply from deployed systems operating at million-scale throughput, where running all candidates per request is prohibitively expensive and incompatible with rapid iteration cycles (Chapelle and Li, 2011; Joachims et al., 2017). In practice, outcomes of unchosen models remain counterfactual and unavailable for learning. Existing benchmarks also frequently use monetary costs as efficiency proxies (Dean and Barroso, 2013), neglecting latency. Because natural dialogue relies on ∼200ms inter-turn gaps (Stivers et al., 2009), multi-second delays break conversational fluidity, so evaluation metrics must ground efficiency in stable, hardware-level characteristics.

To bridge these gaps, we introduce **RouterWorld**, a comprehensive benchmark that aligns LLM routing research with the operational realities of conversational AI. RouterWorld consolidates 53k single-turn core tasks and 17k multi-turn agentic interactions across 7 open-weight LLMs

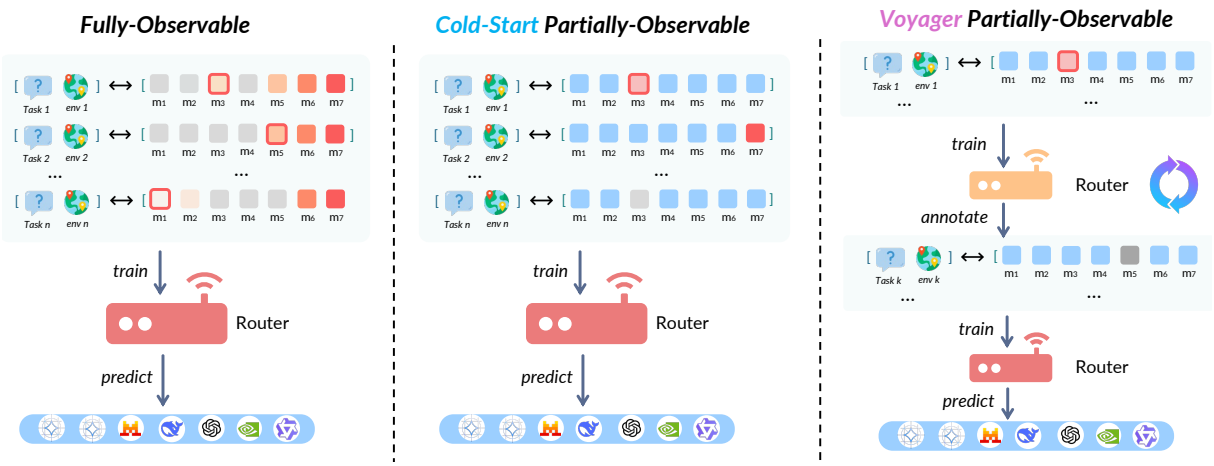

Figure 1: **Routing observability taxonomy.** (*Left*) **Fully-Observable** routing, where feedback from all candidate models is available. (*Center*) **Cold-Start Partially-Observable** routing, where the router observes outcomes only for selected models. (*Right*) **Voyager Partially-Observable** routing, where routing decisions affect future data collection and training. m_i refers to i^{th} candidate model. denotes models not selected by the router, denotes incorrect outputs, while vivid colors indicate correct predictions. Color intensity reflects inference latency (darker indicates higher latency), and dark borders mark optimal models that are both correct and fastest.

(2B–235B parameters, covering coding, math, reasoning, and chit-chat specialists), for a total of **371k** single-turn responses and **76k** multi-turn trajectories once every candidate model is run on every task. To reflect deployed input–output characteristics, we replace artifact-heavy task prompts with natural **Conversational-Form Queries (CF-Queries)**, unify heterogeneous metrics into a binary correctness signal via a standardized LLM-as-a-judge protocol aligned with human evaluation, and replace volatile commercial pricing with reproducible latency metrics obtained from the GenAI-Perf[1] toolkit on controlled hardware.

Crucially, RouterWorld models the counterfactual nature of routing feedback through two partially observable evaluation paradigms (Figure 1). The *Cold-Start* setup simulates zero-shot initialization, training routers from static historical logs without access to unselected model outcomes. The *Voyager* mode extends this to a dynamic, life-long learning setting in which routing decisions shape future training data, exposing the system to non-stationary traffic and testing its ability to mitigate self-reinforcing bias through exploration.

Benchmarking 13 routing strategies on RouterWorld reveals a pronounced efficiency gap. To jointly measure routing quality and efficiency, we introduce **L-DCG**, a latency-aware variant of Discounted Cumulative Gain that weighs each correct selection by how close its latency is to the fastest correct model (formal definition in Section 4.4). Even under idealized full observability, the strongest baseline reaches only 39.6% latency-penalized accuracy (Acc@1-L) compared to the 68.0% empirical upper bound, capturing just 62% of the optimal efficiency ceiling as measured by L-DCG, indicating substantial headroom in latency-aware routing. This gap widens substantially under more realistic partial observability: deprived of counterfactual feedback, routers usually over-emphasize task accuracy (Acc@1) at the expense of latency, inflating accuracy by an average of 24.7% while overall efficiency deteriorates sharply. In the most extreme case, Avenger (Zhang et al., 2025b), a cluster-voting router, collapses into a degenerate *Route-to-Largest* policy (Lai and Ye, 2026), achieving 89.1% Acc@1 but suffering a 79% reduction in optimal efficiency. These feedback-induced failures persist in the Voyager setting, where standard routers show little improvement across streaming interaction, while bandit-based methods with explicit exploration, such as BARP (Wei et al., 2025), continue to improve, highlighting observation diversity. These findings motivate the next generation of router design.

Further, we provide analyses to further clarify routing behavior. We examine long-horizon agentic tasks, where policy collapse emerges through cumulative latency effects rather than single-step decisions (Section 5.3), and study warm-up ini-

[1] https://docs.nvidia.com/deeplearning/triton-inference-server

tialization, finding that most routers function as policy distillers rather than independent optimizers (Section 5.4). By open-sourcing ROUTERWORLD-Core and ROUTERWORLD-Agentic[2], we provide a scalable, deployment-aligned environment for diagnosing these failures and advancing robust, latency-aware routing for conversational AI.

2 Related Work

Non-Predictive and Sequential Routing. Early LLM routing relied on non-predictive orchestration, where decisions are resolved through iterative execution rather than a single selection step. FrugalGPT (Chen et al., 2024) formalizes this via the LLM cascade, escalating queries to progressively stronger models when a confidence threshold is not met. More recent work generalizes this into ensemble-like frameworks: AutoMix (Aggarwal et al., 2024) formulates routing as a POMDP with self-verification signals, and Router-R1 (Zhang et al., 2025a) instantiates the router as a reasoning LLM that plans multi-step invocations. While flexible, these approaches invoke multiple models per query, leading to compounded inference latency, incompatible with strict deployment budgets.

Predictive Routing and Online Model Selection. Predictive approaches formulate model selection as a single decision made prior to generation. HybridLLM (Ding et al., 2024) trains a BERT-based classifier to route between a small and a large model, while RouteLLM (Ong et al., 2025) and Carrot (Tsiourvas et al., 2025) learn from preference or reward signals. GraphRouter (Feng et al., 2025b) models query–model interactions via graph-based message passing, BaRP (Wei et al., 2025) formulates routing as learning from bandit feedback using REINFORCE-style policy gradients, and Avenger (Zhang et al., 2025b) employs train-free cluster-based voting over embedding neighborhoods. IPR (Feng et al., 2025a) targets production settings with user-controlled quality–cost trade-offs, and PILOT (Panda et al., 2025) explores online contextual bandits with exploration–exploitation schedules tailored to routing.

Fully-Observable Routing. Despite this variety of methods, almost all predictive routers are designed and evaluated under *full observability*: at training time, the learner has access to the outcome of every model on every query, reducing routing to supervised learning over complete labels. This assumption underlies RouteLLM (Ong et al., 2025), HybridLLM (Ding et al., 2024), GraphRouter (Feng et al., 2025b), Carrot (Tsiourvas et al., 2025), Avenger (Zhang et al., 2025b), and the Best-of-N style oracles commonly reported as upper bounds (Hu et al., 2024; Huang et al., 2025). While convenient, full observability requires running every model per query, an operational pattern that does not scale to production-scale conversational deployments which preclude exhaustive parallel inference (Chapelle and Li, 2011; Joachims et al., 2017). It also obscures feedback-loop dynamics (confirmation bias, policy collapse, self-reinforcement) that only emerge when the router's choices shape the data it subsequently learns from. A few methods acknowledge partial feedback, notably BaRP (Wei et al., 2025), but its evaluation still assumes rich offline supervision. Our benchmark makes partial observability a first-class evaluation regime and exposes how existing routers behave when counterfactual outcomes are not available.

Router Benchmarks. Routing methods are typically compared using benchmarks such as RouterBench (Hu et al., 2024), RouterArena (Lu et al., 2026), RouterEval (Huang et al., 2025), and LLM-RouterBench (Li et al., 2026), which release precomputed outcomes for many query–model pairs. These resources assume full observability and either omit latency or treat it as a coarse boundary constraint rather than a primary optimization objective. They also rely on raw benchmark prompts and monetary pricing as efficiency proxies, both of which introduce artifacts and drift that are ill-suited to conversational deployment. ROUTERWORLD instead (i) formalizes partial observability via the Cold-Start and Voyager setups, (ii) grounds efficiency in hardware-measured latency that is stable across pricing changes, (iii) adopts natural conversational-form queries, and (iv) spans both single-turn and long-horizon agentic tasks, offering a more deployment-aligned evaluation.

3 Problem Definition

3.1 LLM Routing

Let $\mathcal{M} = \{m_1, \ldots, m_K\}$ denote a fixed set of candidate LLMs. A routing policy

$$\pi : \mathcal{S} \to \mathcal{M}$$

maps an observed state $s \in \mathcal{S}$ to a selected model $m \in \mathcal{M}$. The state s represents the full or par-

[2] RouterWorld-Core and RouterWorld-Agentic datasets.

tial input available to the router. After selecting a model, the router observes a signal reflecting task performance and computational efficiency. In this work, we focus on latency-derived efficiency (Section 4.3) rather than monetary cost. The routing objective is to learn a policy π that maximizes the *expected utility* of the system under these performance and latency constraints.

3.2 Fully-Observable LLM Routing

In the fully-observable setting, the router has access to outcomes of all candidate models for each input during training, reducing routing to supervised learning over complete labels. This has been the dominant assumption in the LLM routing literature to date, underlying most predictive routers and benchmarks surveyed in Section 2 (e.g., Ding et al., 2024; Ong et al., 2025; Feng et al., 2025b; Tsiourvas et al., 2025; Zhang et al., 2025b; Hu et al., 2024; Huang et al., 2025). It is convenient for offline analysis but relies on counterfactual outcomes that are typically unavailable in practice without costly offline replay across all candidate models.

3.3 Partially-Observable LLM Routing

We propose a more realistic scenario of partial observability, where the router observes feedback *only* for the selected model. Formally, after choosing $m = \pi(s)$, the router observes $r(s, m)$, but the counterfactuals $\{r(s, m')\}_{m' \neq m}$ remain missing. The learner must contend with incomplete information. We study two modes of partial observability:

Cold-Start Routing. The Cold-Start setting represents the initialization phase of a routing system. The router must learn from a static historical log generated by a uniform random policy, where each task yields a single model outcome (response, latency, and correctness). This evaluates the router's ability to extract a viable policy from high-entropy, incomplete data in a single offline training round. We explore other warm-up policies in Section 5.4.

Voyager Routing. In real-world deployments, routing decisions are made over streaming inputs, and the router's own policy determines which model outcomes are observed. *Voyager* mode captures this: training proceeds in iterative stages over temporally ordered data chunks $\{C_1, \ldots, C_T\}$. The process is initialized with C_1 (ground-truth feedback under an initial random policy); at stage t, the policy is trained on $\{C_1, \ldots, C_t\}$ and used to route C_{t+1}, with checkpoint selection on a held-out development set. To keep evaluation reproducible without live model inference, we simulate deployment via a *hidden fully-observable engine* derived from ROUTERWORLD: it stores outcomes for all candidates but reveals only the feedback corresponding to the router's selected action.

4 Benchmark Construction

4.1 Dataset Selection Principles

We construct ROUTERWORLD according to four guiding principles: (1) **Task Diversity and Balance**, spanning six major categories (e.g., math, code, reasoning) to prevent domain skew; (2) **Discriminative Power**, selecting examples where model performance varies to necessitate intelligent routing; (3) **Latency Awareness**, integrating real-world timing metrics (Yu et al., 2022; Kwon et al., 2023) to evaluate trade-offs; and (4) **Reproducibility**, enforcing strict train-test separation and automatic metrics. Table 1 compares ROUTERWORLD with existing benchmarks. Unlike prior work, our benchmark uniquely combines partial observability, streaming evaluation, and agentic workflows; see Appendix A for a full breakdown.

4.2 RouterWorld Benchmark Construction

ROUTERWORLD comprises two complementary subsets. **ROUTERWORLD-Core** contains single-turn, end-to-end tasks assessing a router's ability to select appropriate models for standalone queries of varying difficulty. It integrates 12 established benchmarks covering logical and commonsense reasoning (Zellers et al., 2019; Clark et al., 2018; Thorne et al., 2018; Yang et al., 2018), math (Cobbe et al., 2021; Lu et al., 2023; Hendrycks et al., 2021b) and tool use (Patil et al., 2024; Fan et al., 2025), code generation (Hendrycks et al., 2021a; Li et al., 2024), data analysis (Li et al., 2024; Lu et al., 2023), and open-ended chit-chat (Yamashita et al., 2023). **ROUTERWORLD-Agentic** focuses on long-horizon, multi-turn agentic tasks that require environment interaction, covering computer use, household assistance, and e-commerce workflows. All benchmarks are standardized into a unified routing evaluation framework; full preprocessing and standardization procedures are in Appendix C.

ROUTERWORLD Characteristics. Figure 2 and Table 2 summarize the statistics. ROUTERWORLD-CORE contains 48,232 single-turn instances across

Benchmark	#Instances	#Models	#Datasets	#Base.	Part. Obs.	Stream.	Latency Ann.	Agentic Tasks	CF-Query
RouterBench (Hu et al., 2024)	405K	11	8	4	✗	✗	✗	✗	✗
RouteLLM (Ong et al., 2025)	—	2	3	4	✗	✗	✗	✗	✗
LLMRouterBench (Li et al., 2026)	400K	33	21	10	✗	✗	✗	✗	✗
RouterEval (Huang et al., 2025)	200M	8,500+	12	5	✗	✗	✗	✗	✗
RouterArena (Lu et al., 2026)	8.4K	42	23	14	✗	✗	✗	✗	✗
RouterWorld-Core	53K	7	12	17	✓	✗	✓	✗	✓
RouterWorld-Agentic	17K	3	3	17	✓	✓	✓	✓	✓

Table 1: Comparison of LLM routing benchmarks. "Part. Obs." = partially observable setting; "Stream." = streaming rerouting; "Latency Ann." = per-instance latency annotations; "#Base." = number of routing baselines evaluated; "CF-Query" = user query normalization as router input.

seven task categories and twelve datasets, with substantial variation in query difficulty and latency that creates meaningful routing opportunities. RouterWorld-Agentic comprises 12,740 multi-turn interactions in three agentic domains and exhibits higher selectivity and longer latency tails. Analysis of task distributions, difficulty, model specialization, and oracle behavior is in Appendix E.1.

4.3 Annotation

Model Selection & Latency. We select 7 popular open-weight models (2B–235B parameters), creating a heterogeneous landscape of capability and inference cost, where latencies range from ∼200ms to over 7s. We model E2E latency for query-model pairs (q, m) using empirically measured Time-To-First-Token (TTFT_m) and Inter-Token Latency (ITL_m):

$$\text{latency}(q, m) = \text{TTFT}_m + (N_{q,m} - 1) \times \text{ITL}_m, \quad (1)$$

where $N_{q,m}$ is the number of tokens generated. For multi-step agentic trajectories, latency is accumulated across all generation steps. Since raw response and trajectory tokens are released for each task, the framework can be easily adapted to monetary cost. Full profiling workflow and model specifications are in Appendices D.1 and D.

Dataset Evaluation. Source datasets employ heterogeneous evaluation metrics, and individual models often require distinct prompting and post-processing pipelines. To streamline the setup and comparison while mitigating biases introduced by dataset-specific metrics and prompt engineering, we adopt an *LLM-as-a-Judge* framework that produces a standardized binary judgment (*correct* or *incorrect*) for each model output via a strong LLM, i.e. `Claude-4.5-Sonnet`. This binary signal is intended to approximate coarse-grained user feedback in interactive systems, where users typically either adopt a satisfactory response or reject it to ask a follow-up query, rather than providing dense or calibrated rewards (Amershi et al., 2019; Ouyang et al., 2022). We validate this protocol through a blind expert annotation study on 150 (query, model, response) triples: six PhD-level annotators achieve high inter-annotator agreement (Fleiss' κ=0.84, Krippendorff's α=0.87), and the LLM judge matches the 6-expert majority-vote consensus at κ=0.84 (F1=0.93, 92% agreement), approaching the human ceiling and consistent with prior LLM-judge evaluations (Zheng et al., 2023a). Full validation details are provided in Appendix H.

Conversational-Form Query Generation. Real user inputs to voice assistants are typically brief and goal-oriented rather than carefully structured prompts. Further, since benchmarks have been publicly accessible for years, downstream LLMs are susceptible to data contamination and may have memorized surface-level queries and prompt formats during pre-training (Tseng et al., 2025; Golchin and Surdeanu, 2024; Sainz et al., 2023), artificially inflating performance and introducing spurious signals that mislead routers. We therefore apply a multi-stage transformation pipeline: original prompts establish task specifications; a strong external LLM (`Claude-4.5-Sonnet`, excluded from the routing pool) rewrites each as a natural user query; rewritten queries are validated against the original specs, with semantically drifted ones filtered out (Li et al., 2025; Zheng et al., 2023b; Wang et al., 2023; Madaan et al., 2023).

4.4 Metrics

Accuracy@1 and Accuracy@1-L (Optimal). We report two top-1 accuracy metrics for routing. *Accuracy@1* measures whether the model selected by the router correctly answered the query. This metric captures the router's ability to select any correct model, independent of efficiency.

Accuracy@1-L is a stricter, latency-aware vari-

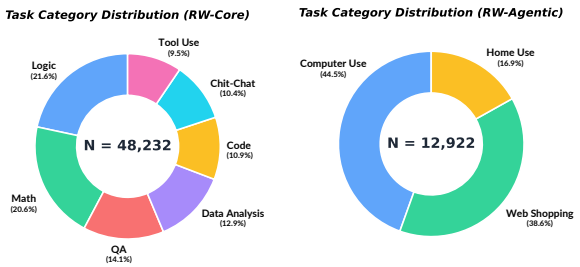

Figure 2: Task category distribution in ROUTER-CORE and ROUTER-AGENTIC.

ant: it equals 1 if the router selects the fastest correct model for the query (i.e., the minimum-latency member of $\mathcal{C}_q$) and 0 otherwise, including when the selected model is incorrect. This metric corresponds to the optimal routing decision under latency constraints and serves as a reference for evaluating latency-sensitive routing performance.

DCG and L-DCG. To evaluate rank-aware routing quality beyond top-1 accuracy, we adopt Discounted Cumulative Gain (DCG). In our setting, relevance is binary: a model is relevant if and only if it produces a correct answer.

For a query q, let $\mathcal{C}_q$ denote the set of correct models, ordered by ascending latency. If the router selects model m^* with rank r in this ordering, the DCG score is defined as

$$\mathrm{DCG}(q) = \frac{\mathbf{1}[m^* \in \mathcal{C}_q]}{\log_2(r+1)}. \tag{2}$$

While DCG captures sensitivity to rank among correct models, it does not reflect the magnitude of latency differences. To incorporate efficiency, we define a latency-aware variant, **L-DCG**:

$$\text{L-DCG}(q) = \mathbf{1}[m^* \in \mathcal{C}_q] \cdot \frac{\ell_{\min}}{\ell_{m^*}} \cdot \frac{1}{\log_2(r+1)}, \tag{3}$$

where $\ell_{\min}$ is the latency of the fastest correct model and ℓ_{m^*} is the latency of the selected model; the score is 0 when the selected model is incorrect. The ratio $\ell_{\min}/\ell_{m^*} \in (0, 1]$ continuously penalizes correct but suboptimal latency choices, reflecting efficiency loss relative to the optimal decision. All metrics are reported as averages over the test set.

5 Experiments

5.1 Experimental Setup

Baselines. We benchmark 13 popular routing algorithms across categories. (1) **Naive routers** provide reference bounds, including *Random* selection, *Always-Route-To-Largest* (always picking the largest candidate model, regardless of query) and *Always-Route-To-Smallest* (always picking the smallest), as well as two oracle upper bounds: *Oracle (Perfect)*, which selects the optimal downstream model with full knowledge of all candidate outcomes, and *Oracle (Best-of-N)*, which, for each query, picks the best downstream model among the N models that at least one evaluated router selects on that query, representing an empirical (rather than theoretical) upper bound attainable by any router in our suite. (2) **Representation-based routers** learn a query representation and route based on similarity, classification, or learned utility in embedding space. This category covers classical methods operating on frozen embeddings (*KNN*, *MLP*), end-to-end representation fine-tuning (*Emb-FT*), and more structured or hybrid architectures such as *Hybrid-LLM* (Ding et al., 2024), *BaRP* (Wei et al., 2025), *GraphRouter* (Feng et al., 2025b), as well as *Avenger* (Zhang et al., 2025b). (3) **Generative routers** use autoregressive language models to directly select a target model via prompting or supervised fine-tuning on routing data. Detailed architectures and training configurations for all baselines are provided in Appendix G.

Statistic	CORE	AGENTIC
Total Instances	48,232	12,740
# Training	40,265	11,292
# Testing	7,967	1,448
# Task Categories	7	3
# Datasets	12	3
# Models	7	6
Prompt len. (mean/max)	2.1k/31k	1.1k/6.3k
Query len. (mean)	110	111
Avg # correct models	4.19	2.95

Table 2: Data characteristics of ROUTERWORLD.

Observability Settings. We evaluate all methods under two regimes: (1) **Fully Observable**, where the training set contains ground-truth correctness and latency for every candidate model; and (2) **Partially Observable**, where each query is annotated with the outcome of only *one* random model.

Implementation details. All experiments are conducted on 8× NVIDIA H200 GPUs. Since most routing algorithms require a query encoder,

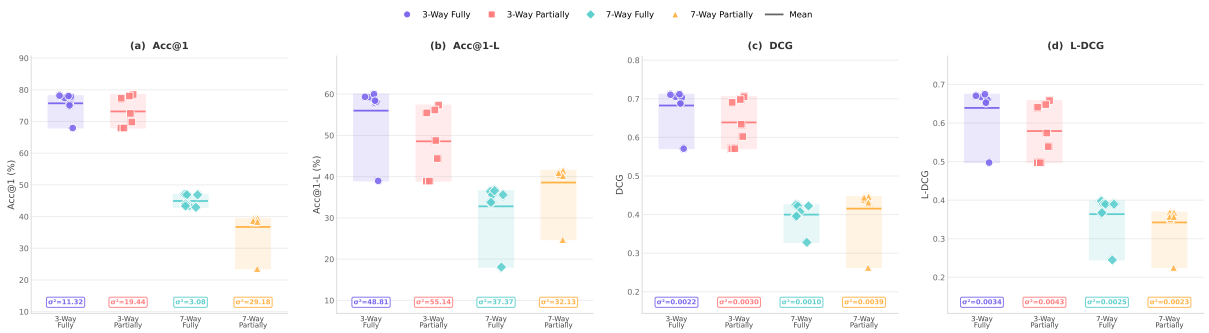

Figure 3: Performance compression in agentic routing. Each dot is a non-collapsed routing baseline. Under both 3-model and 6-model pools, methods cluster tightly with low variance across all metrics. Boxes show interquartile range; σ^2 values shown below.

we standardize on `Qwen3-Embedding-0.6B` (1024-dim, mean-pooled, max sequence length 512) across all methods for a fair comparison. For methods that require architecture-specific encoders (such as Hybrid-LLM which relies on DeBERTa's `[CLS]` token representation) we retain their original encoder as prescribed by the respective papers.

5.2 Main Results

RouterWorld is Challenging. RouterWorld poses a substantial challenge for existing routers (Table 3). Under full observability, the empirical upper bound *Oracle (Best-of-8)* achieves 67.97% Acc@1-L / 0.7140 L-DCG, whereas the strongest non-oracle baseline *Gen-SFT* (Qwen-2.5-1.5B-ins) reaches only 39.61% Acc@1-L / 0.4439 L-DCG, leaving a large gap in latency-aware routing. Effective routing requires context-aware decision-making rather than coarse model attributes, as *Route-To-Largest* attains 89.11% Acc@1 but just 9.10% Acc@1-L

Transitioning to partial observability drives a drastic shift: on average, Acc@1 inflates by 24.7 pp while L-DCG drops by over 0.16. Without cross-model comparisons, routers cannot determine whether a faster model would have also been correct, and become risk-averse (favoring accuracy at the cost of latency). In the extreme, Avenger collapses into Route-To-Largest (89.11% Acc@1, 0.1494 L-DCG), losing any ability to navigate efficiency trade-offs, a fundamental challenge that RouterWorld explicitly captures.

Training Corpus is Important for Routers. To illustrate the necessity of domain-specific training data, we probe Generative AI Routers. Zero-shot prompting with explicit latency profiles severely underperforms (Qwen-2.5-0.5B-Instruct: 7.38% Acc@1-L, 0.1211 L-DCG), while SFT on the RouterWorld corpus drives Acc@1-L to 37.55% and L-DCG to 0.4282. This shows that routing cannot be reduced to a linear relationship between model size and query difficulty: in ∼27% of cases, lightweight models like `Gemma-2-2b` exhibit verbose generation patterns that yield higher latency than larger models. LLM routing is a complex, non-linear decision-making process that requires internalizing empirical interaction logs.

5.3 Routing in Agentic Tasks

We further evaluate routing baselines on RouterWorld-Agentic, where models interact with environments over multiple steps across three benchmarks with new 3-model and 6-model candidate pools (14B–235B). Figure 3 (see Appendix B) shows a striking phenomenon: routing baselines exhibit severe performance compression. In both pools, non-collapsed methods cluster with low variance ($\sigma^2 = 0.81$ for Acc@1-L in the 3-model fully-observable setting; $\sigma^2 = 4.35$ in the 6-model setting; vs. $\sigma^2 = 29.5$ in RouterWorld-Core), and compression intensifies under partial observability, where multiple routers collapse into degenerate near-single-model policies. We attribute this to a factor overlooked by current routing paradigms: the impact of planning steps. High-capability models often resolve complex tasks in far fewer interaction steps than weaker but faster models, neutralizing or even inverting the conventional accuracy–latency trade-off and leaving no meaningful Pareto frontier to exploit. Per-benchmark breakdowns and planning–latency analysis are in Appendix I.

5.4 Analysis of Partially-Observable Routing

We further analyze partially-observable routing: Voyager learning, and warm-up initialization.

Method	Base Model	Fully-Observable				Partially-Observable			
		Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG	Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG
Naive Baselines									
Oracle (Perfect)	–	100.00	100.00	1.0000	1.0000	100.00	100.00	1.0000	1.0000
Oracle (Best-of-8)	–	80.04	67.97	0.7521	0.7140	96.81	54.51	0.7901	0.6462
Random	–	47.63	12.80	0.2995	0.1784	47.63	12.80	0.2995	0.1784
Route-To-Largest	–	89.11	9.10	0.4731	0.1494	89.11	9.10	0.4731	0.1494
Route-To-Smallest	–	33.16	26.55	0.3054	0.2942	33.16	26.55	0.3054	0.2942
Generative AI Routers									
Zero-Shot Prompt	Qwen-2.5-0.5B-ins	**78.65**	7.38	0.3979	0.1211	78.65	7.38	0.3979	0.1211
Gen-SFT	Qwen-2.5-0.5B-ins	55.11	37.55	0.4767	0.4282	81.15	16.91	0.4935	0.2507
Zero-Shot Prompt	Qwen-2.5-1.5B-ins	43.89	10.79	0.2912	0.1953	43.89	10.79	0.2912	0.1953
Gen-SFT	Qwen-2.5-1.5B-ins	54.17	**39.61**	0.4817	**0.4439**	84.55	15.87	**0.4959**	0.2268
Representation-based Routers									
KNN	Qwen3-Embedding-0.6B	52.25	34.35	0.4456	0.3969	71.82	14.86	0.4317	0.2215
MLP	Qwen3-Embedding-0.6B	51.99	38.20	0.4631	0.4295	83.56	14.59	0.4822	0.2158
Emb-FT	Modern-base-0.1B	53.26	33.41	0.4490	0.4041	75.10	15.28	0.4533	0.2342
Emb-FT	Qwen3-Embedding-0.6B	55.93	37.87	0.4833	0.4357	78.57	16.10	0.4702	0.2288
Hybrid-LLM	DeBERTa Large	47.92	35.95	0.4309	0.4039	66.93	**19.52**	0.4352	**0.2661**
Graph-Router	Qwen3-Embedding-0.6B	38.17	13.64	0.2674	0.1994	58.87	13.48	0.3590	0.1892
BARP	Qwen3-Embedding-0.6B	76.58	21.53	**0.4966**	0.2978	**88.57**	9.66	0.4748	0.1565
Avenger	Qwen3-Embedding-0.6B	45.86	32.43	0.4026	0.3709	89.11	9.10	0.4731	0.1494

Table 3: Performance on ROUTERWORLD-Core under fully- and partially-observable settings. Oracle (Perfect) and Oracle (Best-of-8) denote theoretical and empirical upper bounds, respectively. Orange cells indicate collapsed routing that always selects a single model and are excluded from best/second-best comparisons. All pairwise improvements of the best-performing method over the second-best baseline are statistically significant ($p < 0.05$).

Cold-Start v.s. Voyager The Voyager setting incrementally accumulates streaming data over five rounds. Surprisingly, this yields only marginal improvements for most methods (Figure 5, see Appendix B): KNN and MLP L-DCG remains effectively flat, while only *BaRP* shows a noticeable gain ($0.294 \rightarrow 0.320$), consistent with its REINFORCE-based formulation. Supervised methods suffer from confirmation bias under partial observability, reinforcing their initial blind spots by training on their own past decisions; policy-gradient methods instead use observed feedback as an unbiased reward estimator and naturally explore alternatives. Collecting more data under the same biased policy is thus insufficient to mitigate distribution shift: observation *diversity* matters more than volume.

Warm-Up Initialization. To study how initial policy quality influences downstream learning, we simulate three warm-up oracles of increasing sophistication: Qwen-1.5B-Instruct, Claude, and Claude augmented with RAG retrieval from the dev set. Each rolls out selections to generate warm-up logs, on which we then train routers under cold-start, partially-observable conditions. As Figure 4 (see Appendix B) shows, downstream routers correlate near-perfectly with the metric distribution of each warm-up policy: no method surpasses the oracle's own L-DCG, and the ordering of Acc@1, Acc@1-L, DCG, and L-DCG is preserved. Current predictive routing thus largely distills the logging oracle's biases rather than discovering an independent policy, motivating exploration–exploitation strategies that actively seek diverse observations.

6 Conclusion

We introduced ROUTERWORLD, a benchmark that unifies 371k single-turn responses and 76k multi-turn agentic trajectories to evaluate LLM routing under realistic deployment constraints. RouterWorld addresses key gaps in existing benchmarks by formalizing partial observability through Cold-Start and Voyager settings, grounding efficiency in hardware-measured latency rather than volatile pricing, and adopting natural conversational-form queries. Evaluating 13 routing baselines reveals large headroom for latency-aware routing even under idealized conditions. Our analysis highlights the critical importance of partial observability as a design constraint. By open-sourcing all data, code, and evaluation infrastructure, RouterWorld to serve as a foundation for advancing efficiency-aware LLM routing in scalable conversational AI.

Limitations

Multi-modal Tasks. ROUTERWORLD currently evaluates routing only among text-only LLMs. As foundation models increasingly accept vision and audio inputs, practical routing systems may need to decide both *which* model and *which* modality to invoke. Extending ROUTERWORLD to multimodal candidate pools requires new datasets and evaluation protocols that capture heterogeneous latency and cost profiles across modalities, which we leave for future work.

Closed-source Models. ROUTERWORLD focuses on open-weight LLMs to enable reproducible, hardware-grounded latency measurement, which is central to our latency-aware routing framing. As a result, widely used closed-source models (e.g., Claude, GPT-4o) are not included in the candidate pool. Because we release raw responses, trajectory tokens, and generation metadata for every task–model pair, the benchmark can be extended to closed-source APIs by substituting our hardware-grounded latencies with API-measured latencies or per-token pricing.

Ethical Considerations

ROUTERWORLD is built by aggregating 12 publicly released research benchmarks; we redistribute only data permitted under each source's license and preserve original attribution. By design, the benchmark encourages directing simple queries to small, efficient models rather than defaulting to expensive frontier systems. Widespread adoption of latency-aware routing could meaningfully reduce the energy and compute footprint of large-scale conversational AI, benefiting both service providers and the environment.

At the same time, our findings show that routers trained under partial observability can collapse into degenerate policies that consistently favor a single model, potentially degrading response quality for specific query types or user populations without operators noticing. The Voyager setting we propose is designed to surface exactly these failure modes so that future routing algorithms can be stress-tested before deployment.

References

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 . 2024. Automix: Automatically mixing language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Amazon. 2025. How amazon rebuilt alexa with generative ai. https://www.aboutamazon.com/news/devices/new-alexa-tech-generative-artificial-intelligence. Amazon Devices & Services Blog.

Saleema Amershi, Dan Weld, Mihaela Vorvoreanu, Adam Fourney, Besmira Nushi, Penny Collisson, Jina Suh, Shamsi Iqbal, Paul N. Bennett, Kori Inkpen, Jaime Teevan, Ruth Kikin-Gil, and Eric Horvitz. 2019. Guidelines for human-ai interaction. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA. Association for Computing Machinery.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Advances in Neural Information Processing Systems 24: 25th Annual Conference on Neural Information Processing Systems 2011. Proceedings of a meeting held 12-14 December 2011, Granada, Spain*, pages 2249–2257.

Lingjiao Chen, Matei Zaharia, and James Zou. 2024. Frugalgpt: How to use large language models while reducing cost and improving performance. *Trans. Mach. Learn. Res.*, 2024.

Mahe Chen, Xiaoxuan Wang, Kaiwen Chen, and Nick Koudas. 2025. Generative engine optimization: How to dominate ai search. *arXiv preprint arXiv:2509.08919*.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael I. Jordan, Joseph E. Gonzalez, and Ion Stoica. 2024. Chatbot arena: An open platform for evaluating llms by human preference. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Jeffrey Dean and Luiz André Barroso. 2013. The tail at scale. *Commun. ACM*, 56(2):74–80.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Rühle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. 2024. Hybrid LLM: cost-efficient and quality-aware query routing. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Shiqing Fan, Xichen Ding, Liang Zhang, and Linjian Mo. 2025. Mcptoolbench++: A large scale ai agent model context protocol mcp tool use benchmark. *Preprint*, arXiv:2508.07575.

Aosong Feng, Balasubramaniam Srinivasan, Yun Zhou, Zhichao Xu, Kang Zhou, Sheng Guan, Yueyan Chen, Xian Wu, Ninad Kulkarni, Yi Zhang, Zhengyuan Shen, Dmitriy Bespalov, Soumya Smruti Mishra, Yifei Teng, Darren Yow-Bang Wang, Haibo Ding, and Lin Lee Cheong. 2025a. IPR: Intelligent prompt routing with user-controlled quality-cost trade-offs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*. Association for Computational Linguistics.

Tao Feng, Yanzhen Shen, and Jiaxuan You. 2025b. Graphrouter: A graph-based router for LLM selections. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Shahriar Golchin and Mihai Surdeanu. 2024. Time travel in LLMs: Tracing data contamination in large language models. In *The Twelfth International Conference on Learning Representations*.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. 2021a. Measuring coding challenge competence with apps. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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

Zhongzhan Huang, Guoming Ling, Yupei Lin, Yandong Chen, Shanshan Zhong, Hefeng Wu, and Liang Lin. 2025. RouterEval: A comprehensive benchmark for routing LLMs to explore model-level scaling up in LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, Suzhou, China. Association for Computational Linguistics.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, WSDM '17, page 781–789, New York, NY, USA. Association for Computing Machinery.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*. Association for Computing Machinery.

Guannan Lai and Han-Jia Ye. 2026. When routing collapses: On the degenerate convergence of llm routers. *arXiv preprint arXiv:2602.03478*.

Hao Li, Yiqun Zhang, Zhaoyan Guo, Chenxu Wang, Shengji Tang, Qiaosheng Zhang, Yang Chen, Biqing Qi, Peng Ye, Lei Bai, Zhen Wang, and Shuyue Hu. 2026. Llmrouterbench: A massive benchmark and unified framework for llm routing. *Preprint*, arXiv:2601.07206.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, and 1 others. 2024. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems*, 36.

Jinyang Li, Xiaolong Li, Ge Qu, Per Jacobsson, Bowen Qin, Binyuan Hui, Shuzheng Si, Nan Huo, Xiaohan Xu, Yue Zhang, Ziwei Tang, Yuanshuai Li, Florensia Widjaja, Xintong Zhu, Feige Zhou, Yongfeng Huang, Yannis Papakonstantinou, Fatma Ozcan, Chenhao Ma, and Reynold Cheng. 2025. SWE-SQL: Illuminating LLM pathways to solve user SQL issues in real-world applications. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Nelson F. Liu, Tianyi Zhang, and Percy Liang. 2023. Evaluating verifiability in generative search engines. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*. Association for Computational Linguistics.

Hugo Touvron Llama. 2023. Llama: Open and efficient foundation language models.

Pan Lu, Liang Qiu, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, Tanmay Rajpurohit, Peter Clark, and Ashwin Kalyan. 2023. Dynamic prompt learning via policy gradient for semi-structured mathematical reasoning. In *International Conference on Learning Representations (ICLR)*.

Yifan Lu, Rixin Liu, Jiayi Yuan, Xingqi Cui, Shenrun Zhang, Hongyi Liu, and Jiarong Xing. 2026. Routerarena: An open platform for comprehensive comparison of LLM routers. In *The Fourteenth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E. Gonzalez, M Waleed Kadous, and Ion Stoica. 2025. Routellm: Learning to route llms with preference data. *Preprint*, arXiv:2406.18665.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. 2025. Adaptive LLM routing under budget constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 23934–23949, Suzhou, China.

Shishir G Patil, Tianjun Zhang, Xin Wang, and Joseph E. Gonzalez. 2024. Gorilla: Large language model connected with massive APIs. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. Toolllm: Facilitating large language models to master 16000+ real-world apis. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*.

Oscar Sainz, Jon Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre. 2023. NLP evaluation in trouble: On the need to measure LLM data contamination for each benchmark. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. Association for Computational Linguistics.

Tanya Stivers, Nicholas J Enfield, Penelope Brown, Christina Englert, Makoto Hayashi, Trine Heinemann, Gertie Hoymann, Federico Rossano, Jan Peter De Ruiter, Kyung-Eun Yoon, and 1 others. 2009. Universals and cultural variation in turn-taking in conversation. *Proceedings of the National Academy of Sciences*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. 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)*.

Yuan Tseng, Titouan Parcollet, Rogier van Dalen, Shucong Zhang, and Sourav Bhattacharya. 2025. Evaluation of llms in speech is often flawed: Test set contamination in large language models for speech recognition. *arXiv preprint arXiv:2505.22251*.

Asterios Tsiourvas, Wei Sun, and Georgia Perakis. 2025. Causal LLM routing: End-to-end regret minimization from observational data. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. 2025. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Wang Wei, Tiankai Yang, Hongjie Chen, Yue Zhao, Franck Dernoncourt, Ryan A. Rossi, and Hoda Eldardiry. 2025. Learning to route llms from bandit feedback: One policy, many trade-offs. *Preprint*, arXiv:2510.07429.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2024. Autogen: Enabling next-gen LLM applications via multi-agent conversations. In *First Conference on Language Modeling*.

Sanae Yamashita, Koji Inoue, Ao Guo, Shota Mochizuki, Tatsuya Kawahara, and Ryuichiro Higashinaka. 2023. RealPersonaChat: A realistic persona chat corpus with interlocutors' own personalities. In *Proceedings of the 37th Pacific Asia Conference on Language, Information and Computation*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, and Byung-Gon Chun. 2022. Orca: A distributed serving system for Transformer-Based generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, Carlsbad, CA. USENIX Association.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, Florence, Italy. Association for Computational Linguistics.

Haozhen Zhang, Tao Feng, and Jiaxuan You. 2025a. Router-r1: Teaching LLMs multi-round routing and aggregation via reinforcement learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yiqun Zhang, Hao Li, Chenxu Wang, Linyao Chen, Qiaosheng Zhang, Peng Ye, Shi Feng, Daling Wang, Zhen Wang, Xinrun Wang, Jia Xu, Lei Bai, Wanli Ouyang, and Shuyue Hu. 2025b. The avengers: A simple recipe for uniting smaller language models to challenge proprietary giants. *Preprint*, arXiv:2505.19797.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023a. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023b. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

A Dataset Selection Principles

The construction of ROUTERWORLD is guided by the following design principles to ensure a comprehensive and fair assessment of routing capabilities:

- **Capability Diversity:** We include benchmarks spanning six major task categories. Mathematical reasoning, code generation, structured query generation, commonsense and multi-hop reasoning, factual verification and tool use, and open-ended dialogue. This diversity ensures that no single model dominates across all tasks, making routing decisions non-trivial.

- **Discriminative Power:** Each benchmark is selected to induce heterogeneous performance across the model pool. In particular, smaller or specialized models (e.g., `deepseek-math` on GSM8K or `codegemma` on APPS) may outperform larger general-purpose models on domain-specific tasks. Such performance divergence is essential for revealing the benefits of intelligent routing over static model selection.

- **Latency Awareness:** Beyond answer correctness, we incorporate real-world serving latency measurements, including time-to-first-token and end-to-end response time. This enables evaluation of quality–latency trade-offs that are critical in practical deployment settings.

- **Reproducibility:** We prioritize benchmarks with well-established evaluation protocols and automatic metrics, avoiding subjective human judgments to ensure consistent and reproducible comparisons across routing methods.

- **Balanced Task Distribution:** Unlike prior benchmarks that are often skewed toward specific capabilities, RouterWorld enforces a balanced distribution across task categories. This design mitigates evaluation bias introduced by over-represented task types and enables fair assessment of routing performance across diverse capabilities.

- **Training–Testing Separation:** Our study evaluates both router models and routing algorithms, so we prioritize benchmarks that provide train and test splits. This separation supports principled learning-based routing and prevents information leakage during evaluation.

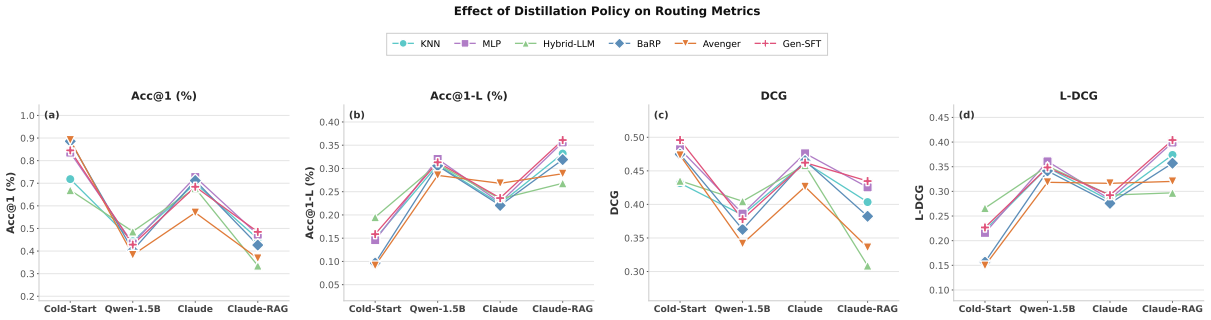

Figure 4: Routing method performance across distillation policies in the partially-observable setting. All methods exhibit a high correlation with the distillation policy trend

B Additional Figures

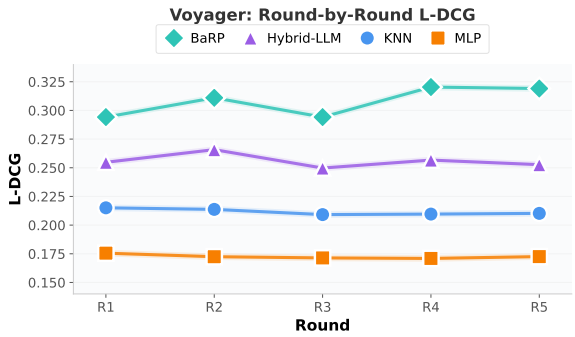

Figure 5: Voyager round-by-round L-DCG.

C Benchmark Dataset Details

The RouterWorld benchmark comprises the following datasets, selected to represent a diverse range of linguistic and symbolic reasoning tasks:

HellaSwag (Zellers et al., 2019). A sentence completion benchmark that challenges models to select the most plausible continuation of a given context. It employs adversarial filtering to construct distractors that are deceptively plausible to language models while remaining clearly incorrect to humans, effectively testing commonsense reasoning.

GSM8K (Cobbe et al., 2021). A dataset of 8.5K linguistically diverse grade-school math word problems. Each problem typically requires 2–8 steps of elementary arithmetic, testing a model's ability to perform reliable sequential calculations.

ARC-Challenge (Clark et al., 2018). The partition of the AI2 Reasoning Challenge containing grade-school science questions that cannot be solved via simple retrieval or pattern matching. These items require deeper reasoning and scientific understanding.

APPS (Hendrycks et al., 2021a). A code generation benchmark comprising 10,000 programming problems collected from open-access platforms like Codeforces. The dataset spans three difficulty levels (introductory, interview, and competition) and requires generating executable Python programs from natural language specifications.

BIRD-SQL (Li et al., 2024). A large-scale, cross-domain text-to-SQL benchmark featuring real-world databases with noisy and incomplete values. It emphasizes schema understanding and database content reasoning over complex relational structures.

FEVER (Thorne et al., 2018). A fact verification dataset containing 185K claims derived from Wikipedia. Models must classify claims as *Supported*, *Refuted*, or *NotEnoughInfo* based on retrieved evidence, testing evidence-grounded factual reasoning.

Gorilla APIBench (Patil et al., 2024). An evaluation of LLM-based API call generation across major hubs (TorchHub, TensorHub, HuggingFace). It tests the ability to produce accurate, executable API invocations from natural language instructions.

HotpotQA (Yang et al., 2018). A multi-hop question answering benchmark requiring reasoning over multiple supporting documents. Questions necessitate combining information from two or more Wikipedia passages, testing compositional reasoning and information synthesis.

PersonaChat (Yamashita et al., 2023). A chit-chat dialogue dataset where each speaker is assigned a specific persona defined by profile sen-

tences. Models must generate coherent, engaging responses consistent with the assigned persona.

TabMWP (Lu et al., 2023). A collection of 38K tabular math word problems combining natural language questions with structured tables. Solutions require extracting relevant numerical information and performing multi-step quantitative reasoning over the tabular data.

Hendrycks-MATH (Hendrycks et al., 2021b). A dataset of 12,500 competition-level mathematics problems across seven domains, including algebra, geometry, and number theory. These problems demand formal, multi-step mathematical reasoning beyond elementary arithmetic.

MCPToolBench++ (Fan et al., 2025). Evaluates tool-use capabilities under the Model Context Protocol (MCP). It tests the model's ability to correctly invoke tools via structured function calls and accurately extract parameters from natural language instructions.

D Model Specifications and Latency

D.1 Latency Profiling Protocol

To obtain per-model latency estimates for the routing objective, we profile the inference latency of all candidate LLMs using NVIDIA GenAI-Perf, a standardized benchmarking tool built on top of Triton Perf Analyzer.

Setup. Each model is served independently via vLLM (Kwon et al., 2023) inside a Docker container with NVIDIA GPU support, using tensor parallelism across 4 GPUs. The benchmarking client runs inside a separate NVIDIA Triton SDK container (`tritonserver:24.06-py3-sdk`) on the same host, communicating with the vLLM server over the OpenAI-compatible chat endpoint. All models are evaluated under identical conditions: 1,000 synthetic prompts with a fixed input length of 2,000 tokens and output length of 1,000 tokens, at a concurrency level of 10, with streaming enabled.

Metrics. We extract two key metrics for each model:

- **Time to First Token (TTFT)**: the average time from request submission to the first generated token, reflecting model loading and prefill latency.
- **Inter-Token Latency (ITL)**: the average time between consecutive output tokens during streaming decoding.

Total latency estimation. For RouterWorld-Core, where each instance is a single-turn query with N output tokens, we estimate the end-to-end generation latency as:

$$\text{Latency}_{\text{core}}(m, N) = \text{TTFT}_m + (N-1) \times \text{ITL}_m \quad (4)$$

where m denotes the model.

For RouterWorld-Agentic, tasks involve multi-step episodes where the agent interacts with an environment over K turns, each producing N_k tokens. The total episode latency is:

$$\text{Latency}_{\text{agent}}(m) = \sum_{k=1}^{K} \left[\text{TTFT}_m + (N_k - 1) \times \text{ITL}_m\right] + \sum_{k=1}^{K} \text{Env}_k \quad (5)$$

where Env_k denotes the environment response time at step k (e.g., browser rendering, API calls). In practice, we record the wall-clock episode latency directly from the agentic harness, which captures both model inference and environment interaction time.

These formulas are used to compute per-instance latency for each model in the routing dataset, enabling the construction of latency-aware ground-truth labels.

D.2 Unified Latency Profile

Table 4 presents the latency profile of all candidate models across both RouterWorld-Core and RouterWorld-Agentic, measured with GenAI-Perf under identical conditions. Models are sorted by total estimated latency (Eq. 4 with N=1,000). The **Pool** column indicates whether the model is used in Core, Agentic, or both benchmarks.

D.3 Model Descriptions

Core-Only Models. **Gemma-2B** serves as the efficiency anchor with a median latency of just 13ms TTFT. **CodeGemma-7B** and **DS-Math-7B** are domain-specialized models that test the router's ability to identify task intent, offering low latency ($\sim$100ms) while outperforming generalists on code and math tasks. **Mistral-7B** provides a robust generalist baseline for the 7B class.

Model	Pool	Params	TTFT (ms)	ITL (ms)	Total (s)
Gemma-2B	Core	2B	12.64	3.96	3.97
CodeGemma-7B	Core	7B	14.22	5.20	5.21
Ministral-14B	Agentic	14B	18.26	3.50	3.51
Mistral-7B	Core	7B	100.21	4.13	4.23
DS-Math-7B	Core	7B	117.16	4.36	4.48
Devstral-123B	Agentic	123B	493.30	14.56	15.04
Qwen3-Coder-30B	Agentic	30B	532.29	7.20	7.73
GPT-OSS-20B	Both	20B	1,137.45	4.06	5.19
GPT-OSS-120B	Agentic	120B	1,212.55	5.97	7.18
Qwen3-235B	Both	235B	2,258.10	18.76	21.01
Nemotron-9B	Core	9B	5,733.59	6.29	12.03
Qwen3-Next-80B	Agentic	80B	20,612.40	8.38	28.99

Table 4: Unified latency profile of all candidate models, measured with GenAI-Perf (2,000 input tokens, 1,000 output tokens, concurrency 10, streaming). Models sorted by total latency.

Agentic-Only Models. **Ministral-14B** is the fastest agentic model (18ms TTFT, 3.5ms ITL). **Qwen3-Coder-30B** represents a mid-tier option balancing latency and accuracy. **Devstral-123B** and **GPT-OSS-120B** are large-scale models for complex agentic tasks. **Qwen3-Next-80B** exhibits the highest TTFT (20.6s) due to cold-start overhead.

Shared Models. **GPT-OSS-20B** bridges both pools, achieving 67.9% accuracy on Core with moderate latency ($\sim$5.2s). **Qwen3-235B** is the oracle model of both pools with the highest accuracy but significant latency ($\sim$21s). **Nemotron-9B** achieves high accuracy (71.8% on Core) but exhibits disproportionately high TTFT (5.7s), presenting a challenging routing trap: high accuracy at high cost.

Infrastructure. All profiling experiments are conducted on a single node equipped with NVIDIA GPUs. Models requiring newer architecture support (Ministral-14B, Qwen3-Next-80B, Devstral-123B) use vLLM v0.12.0, while the remaining models use vLLM v0.9.2, all with tensor parallelism of 4.

E Detailed Dataset Statistics & Analysis

E.1 RouterWorld-Core Statistics

In this section, we analyze the specific difficulty distributions and routing opportunities within RouterWorld-Core and RouterWorld-Agentic.

Composition and Specialization. The RouterWorld-Core test set ($N = 7{,}967$) provides balanced coverage across Reasoning (Logic 17.3%, Math 18.5%, Data Analysis 18.9%) and Interaction (QA 16.3%, Code 12.0%, Tool Use 10.6%, Chit-Chat 6.4%). As shown in Figure 6, no single model dominates all categories. For example, `DS-Math-7B` leads sub-20B models on Math tasks but underperforms on Logic, while `CodeGemma-7B` shows distinct strength in code generation. This confirms that effective routing must account for task-specific competence rather than relying solely on model scale.

Routing Opportunity. Query difficulty is broadly distributed (Figure 7), creating significant opportunity for adaptive routing. Oracle routing selects the *fastest correct* model per query, demonstrating that frontier-scale models are often computationally wasteful for standard tasks (Figure 9).

E.2 RouterWorld-Agentic Statistics

Composition. The RouterWorld-Agentic benchmark ($N_{\text{train}} = 11{,}292$, $N_{\text{test}} = 1{,}448$) covers three agentic task families: Web Navigation (44.8%), Online Shopping (38.2%), and Household (17.0%), drawn from WorkArena, WebShop, and ALFWorld respectively. These tasks require multi-step reasoning, tool use, and environment interaction, presenting fundamentally different routing challenges compared to single-turn Core tasks.

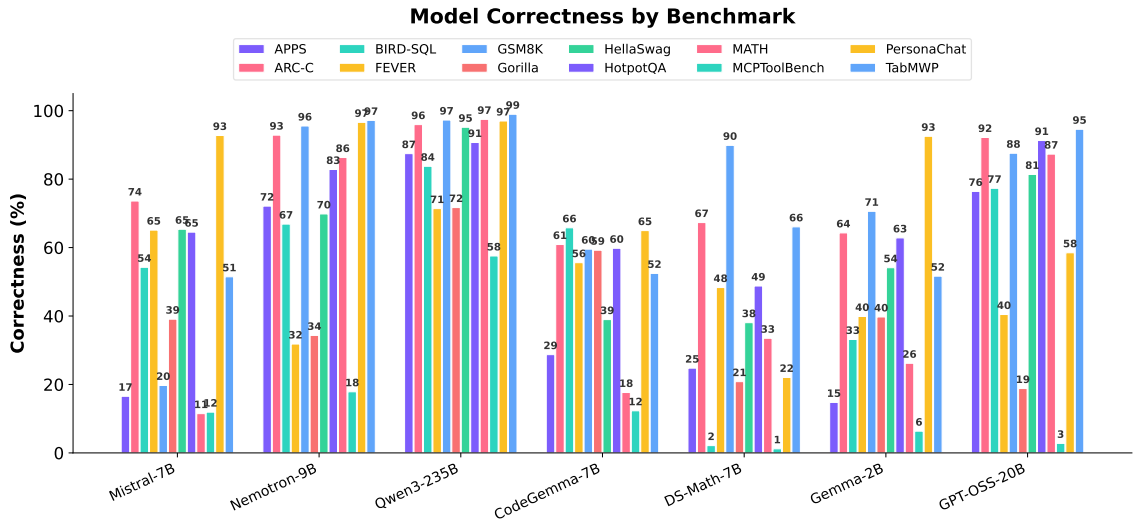

Figure 6: Per-benchmark model correctness for RouterWorld-Core. Models exhibit distinct specialization patterns across task types, with smaller models often outperforming generalists in specific domains.

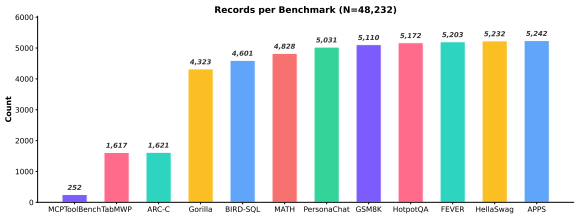

Figure 7: Records per benchmark in RouterWorld-Core. The dataset spans 12 benchmarks with broad coverage across reasoning, coding, and interaction tasks.

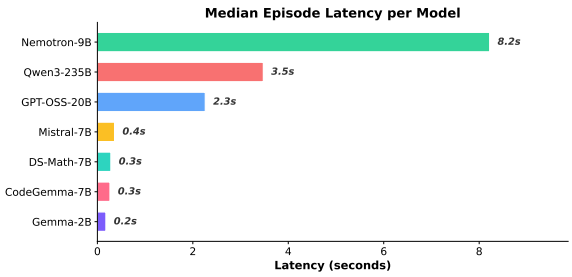

Figure 8: Median episode latency per model in RouterWorld-Core. Smaller models offer substantially lower latency, motivating intelligent routing.

Model Correctness by Benchmark. As shown in Figure 10, model correctness varies substantially across agentic benchmarks. Larger models generally achieve higher correctness on complex web navigation tasks, but smaller specialized models can be competitive on structured household tasks. This heterogeneity across benchmarks reinforces the value of task-aware routing.

Latency and Routing Opportunity. Agentic tasks exhibit significantly higher latency than Core tasks (median 11.8s vs. 4.3s), with large variance across models (Figure 12). This amplifies the potential savings from intelligent routing. Figure 13 shows the fastest correct model distribution, revealing that even in the agentic setting, smaller models frequently provide the fastest correct response.

F Router Implementation

F.1 Hyperparameters

The regression head is implemented as a three-layer Multi-Layer Perceptron (MLP) with hidden dimensions $[512, 256, 128]$. Each linear layer is followed by a ReLU activation, Batch Normalization, and Dropout ($p = 0.3$). The final projection maps the hidden state to K logits. We train using the AdamW optimizer with a learning rate of 2×10^{-5} for the encoder and 1×10^{-4} for the head, employing a cosine annealing schedule with a 5% warmup period.

G Baseline Implementation Details

We implement all baselines using PyTorch and train on 8× NVIDIA H200 GPUs. Below we detail the architecture and hyperparameters for each method under both the *fully observable* and *partially observable* settings. In the fully observable

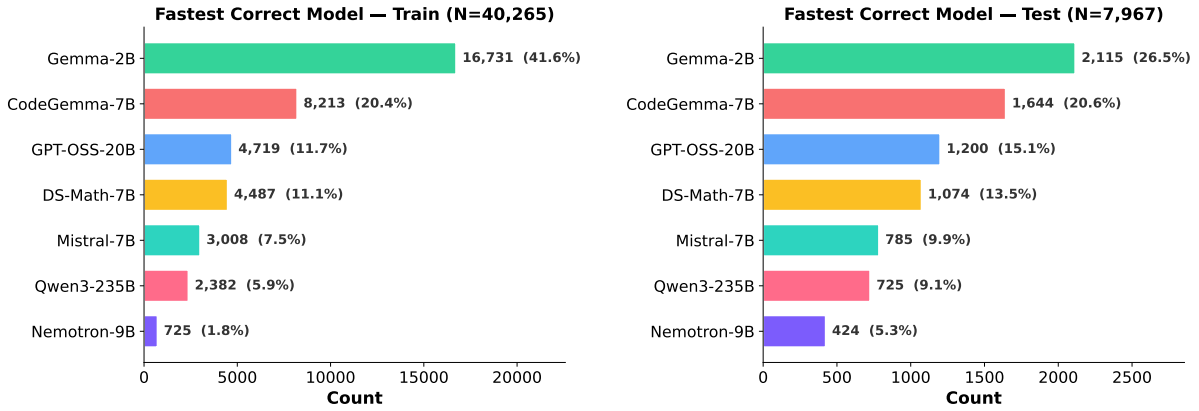

Figure 9: Fastest correct model distribution for ROUTERWORLD-Core (train and test). Small models are the optimal choice for a large share of queries.

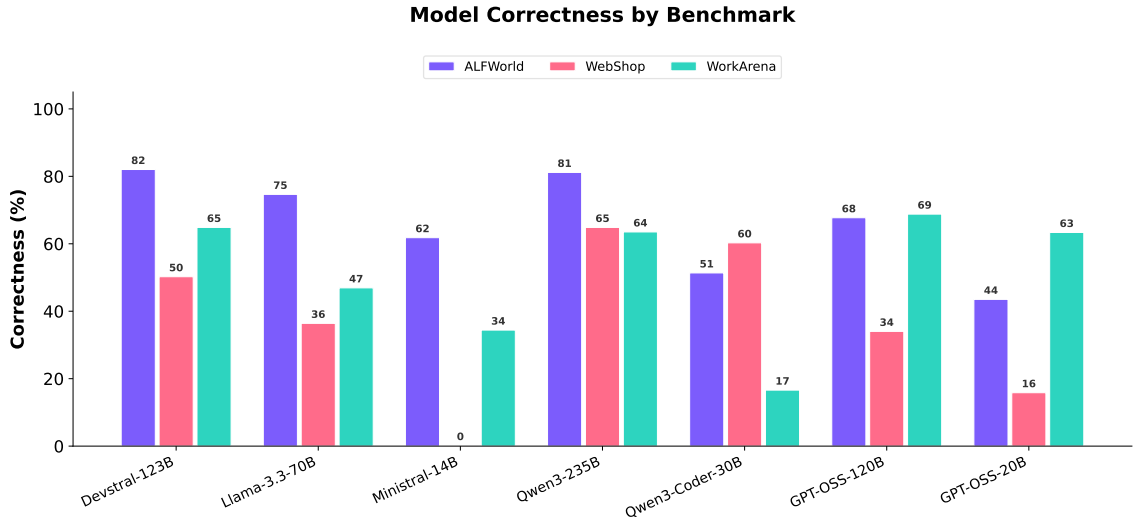

Figure 10: Per-benchmark model correctness for ROUTERWORLD-Agentic. Performance varies substantially across the three agentic task families.

setting, the router has access to the correctness labels of *all* candidate models for each training query (`correct_list`). In the partially observable setting, the router only observes the outcome of a *single* selected model per query (`selected` field), and must learn to route without knowledge of unobserved model outcomes. Unless otherwise noted, all methods share the same architecture across settings; we highlight where hyperparameters or training procedures differ.

G.1 Representation-Based Routers

Representation-based routers learn a query representation and make routing decisions based on similarity, classification, or learned utility in embedding space. These methods differ in whether representations are frozen or fine-tuned, and in how routing decisions are derived from the learned representations.

- **Naive Bayes & KNN (k=5):** Operate on frozen representations from `Qwen3-Embedding-0.6B` (mean-pooled, 1024-dimensional), predicting a distribution over the K candidate models and selecting the argmax. These methods are non-parametric and require no gradient-based training.
  - *Fully observable:* Trained on the full training set with labels derived from `correct_list`.
  - *Partially observable:* Trained on the filtered subset containing only samples where the selected model was correct (`correct_only`).

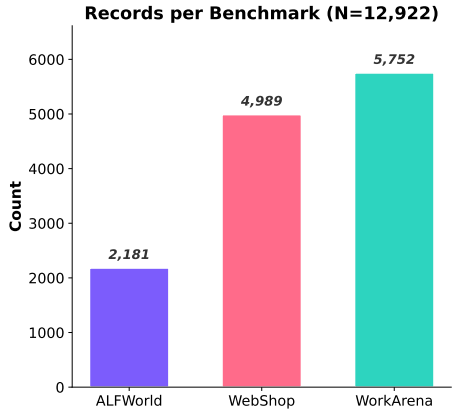

Figure 11: Records per benchmark in RouterWorld-Agentic across three agentic task families.

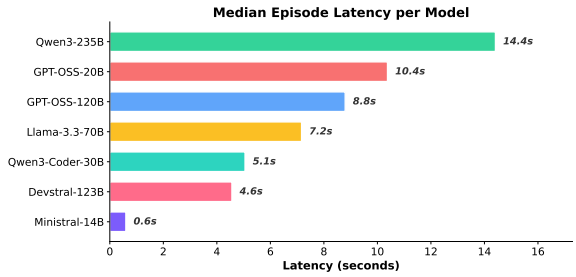

Figure 12: Median episode latency per model in RouterWorld-Agentic. Latency ranges from 0.6s to 16.4s across models.

- **MLP:** A three-hidden-layer feedforward classifier ($1024 \rightarrow 512 \rightarrow 256 \rightarrow 128 \rightarrow K$) trained on frozen `Qwen3-Embedding-0.6B` representations using cross-entropy loss (learning rate 1×10^{-4}, dropout 0.2).
  - *Fully observable:* Trained for 12 epochs on the full training set.
  - *Partially observable:* Trained for 20 epochs on the `correct_only` subset.
- **Emb-FT:** End-to-end fine-tuning of `Qwen3-Embedding-0.6B` with a two-layer classification head (hidden dimension 512, dropout 0.1), trained with DDP across 8 GPUs (learning rate 1×10^{-4}, per-GPU batch size 32, gradient accumulation 2, effective batch size 512).
  - *Fully observable:* 10 epochs on the full training set.
  - *Partially observable:* 10 epochs on the `correct_only` subset, with labels from the `selected` field.
- **ModernBERT** (Warner et al., 2025): Fine-tunes `ModernBERT-base` end-to-end with a two-layer classification head (hidden dimension 768, dropout 0.1) using DDP across 8 GPUs.
  - *Fully observable:* Learning rate 1×10^{-3}, per-GPU batch size 32, gradient accumulation 2, effective batch size 512, 10 epochs.
  - *Partially observable:* Learning rate 1×10^{-3}, per-GPU batch size 128, gradient accumulation 2, effective batch size 2,048, 10 epochs on the `correct_only` subset.
- **Hybrid-LLM** (Ding et al., 2024): Fine-tunes `DeBERTa-v3-large` with an N-class cross-entropy objective (learning rate 5×10^{-5}, dropout 0.1) using DDP across 8 GPUs.
  - *Fully observable:* Per-GPU batch size 32, gradient accumulation 2, effective batch size 512, 20 epochs. Uses `correct_list[0]` (fastest correct model) as the ground-truth label.
  - *Partially observable:* Per-GPU batch size 64, gradient accumulation 2, effective batch size 1,024, 15 epochs on the `correct_only` subset. Uses the `selected` model as the ground-truth label.
- **BaRP** (Wei et al., 2025): A preference-conditioned routing policy trained with REINFORCE on top of frozen `Qwen3-Embedding-0.6B` representations (learning rate 1×10^{-4}, entropy coefficient β=0.05, dropout 0.2). At inference, users specify a preference vector (w_q, w_c) to trade off quality and cost; the policy outputs $\pi(a \mid x, w)$ over candidate models. Trained with DDP across 8 GPUs (per-GPU batch size 32, effective batch size 256).
  - *Fully observable:* Uses expected reward over all models from `correct_list`, up to 100 epochs with early stopping (patience 10).
  - *Partially observable:* Learns only from the outcome of the `selected` model per sample, up to 100 epochs with early stopping (patience 10).

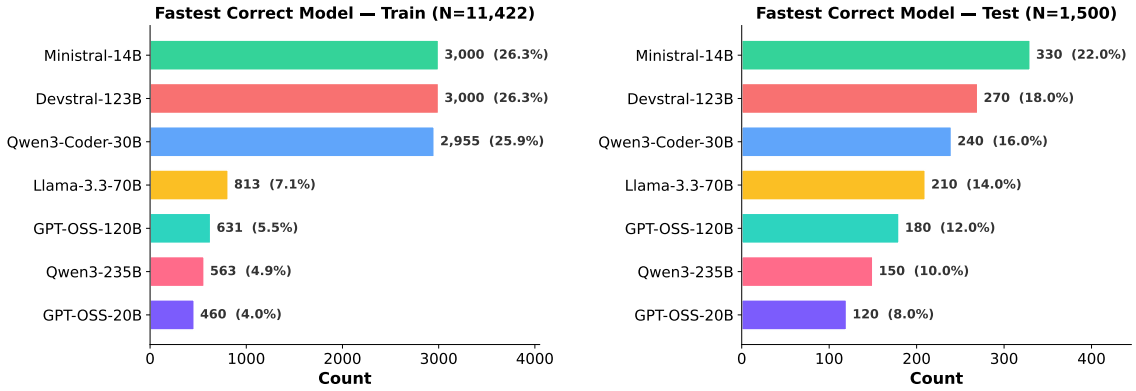

Figure 13: Fastest correct model distribution for RouterWorld-Agentic (train and test).

- **GraphRouter** (Feng et al., 2025b): A graph neural network modeling query–model interactions. GNN layers (hidden dimension 8, edge mask rate 0.5) are trained on top of frozen `Qwen3-Embedding-0.6B` embeddings with a learning rate of 3×10^{-4} and weight decay 1×10^{-4}.
  - *Fully observable:* Batch size 128, 100 epochs, single GPU. Labels derived from `correct_list[0]` (fastest correct model).
  - *Partially observable:* Batch size 128, 1,000 epochs, single GPU. Only the selected model receives gradient (mask $=$ 1); unobserved models are masked out.
- **Avenger** (Zhang et al., 2025b): A *training-free*, cluster-based ranking router. During offline calibration, training queries are embedded with frozen `Qwen3-Embedding-0.6B` and clustered via K-means (k=64). Per-cluster model rankings are computed from the training labels. At inference, the query is assigned to the top-11 nearest clusters (softmax temperature β=3.0), and rankings are aggregated to select the best model.
  - *Fully observable:* Rankings computed using L-DCG (latency-aware discounted cumulative gain) over all models in `correct_list`.
  - *Partially observable:* Rankings computed using `router_accuracy` over the `selected` field only, as latency-aware metrics require observing all models.

G.2 Generative Routers

Generative routers use autoregressive language models to directly select a target model via prompting or supervised fine-tuning on routing data.

- **Zero-Shot Prompting:** We serve `Qwen2.5-0.5B-Instruct` via vLLM with 128 parallel workers. The model receives a system prompt describing the available model set and outputs the selected model name directly. This baseline is identical across both settings as it requires no training data.
- **Gen-SFT:** Supervised fine-tuning of `Qwen2.5-1.5B-Instruct` on routing annotations using standard cross-entropy loss (learning rate 2×10^{-5}, micro batch size 4, effective batch size 128, max sequence length 1,024, weight decay 0.01, 5 epochs, 8 GPUs with FSDP).
  - *Fully observable:* Training targets are the fastest correct model from `correct_list`.
  - *Partially observable:* Training targets are the `selected` model, filtered to correct-only samples.

H LLM Judge Validation Details

We validate the automated LLM judge used throughout RouterWorld against independent human annotations. Six experts blind-label 150 stratified (query, model, response) triples sampled across 12 datasets and 7 models without seeing the judge verdict. We report inter-annotator agreement (IAA) to establish the task-difficulty ceiling, and LLM–human agreement (LHA) to measure judge alignment with expert majority-vote consensus.

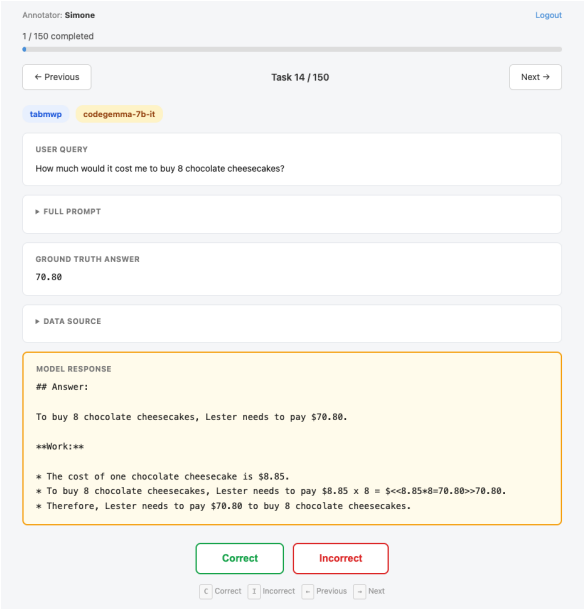

Figure 14: Screenshot of the web-based annotation tool developed for the LLM judge validation study. The interface displays the source dataset and model (colored tags), the user query, the full prompt, the ground-truth answer, and the model response. Annotators label each triple as *correct* or *incorrect* by comparing the model response against the ground truth. The tool and annotation guidelines will be released as open source.

H.1 Annotation Protocol

We developed a web-based annotation tool (Figure 14) that presents each annotator with: (i) the user query, (ii) the full prompt including instructions and response choices, (iii) the ground-truth answer from the dataset, and (iv) the model response. The source dataset and responding model are displayed as colored tags. Annotators follow the guidelines below.

Core Rule. For each triple, the annotator compares the model response against the ground-truth answer and labels it *correct* or *incorrect*.

Key Guidelines.

1. **Focus on the final answer only.** Formatting differences (e.g., `<a>C</a>` vs. `C.`) and intermediate reasoning traces are ignored; only the substantive answer matters.

2. **Compare against ground truth.** Annotators judge whether the model's answer matches the provided ground-truth label, without independently verifying the ground truth itself.

3. **Skip ambiguous cases.** When a confident judgment is not possible—e.g., long code-generation tasks requiring execution, or subjective dialogue evaluations (persona-chat)—annotators skip the entry to preserve annotation integrity.

Metric	Value
Fleiss' κ	0.835
Krippendorff's α	0.867
Avg. pairwise κ	0.892
Avg. pairwise agreement	0.950

Table 5: Inter-annotator agreement among 6 experts on 150 blind annotations.

Comparison	Agreement	κ	Precision / Recall	F1
Consensus vs. LLM Judge	0.920	0.835	0.98 / 0.89	0.932

Table 6: Majority-vote consensus (6 annotators) vs. LLM judge (n=150).

H.2 Agreement Scores

Table 5 reports multi-rater agreement among the six annotators. The high Fleiss' κ=0.84 reflects the largely objective nature of the benchmarks: for datasets with unambiguous answers (e.g., multiple-choice, numeric), annotators almost always concur. Disagreements arise primarily on tasks requiring non-trivial answer extraction (e.g., BIRD-SQL, GORILLA_FUNC).

Table 6 compares the majority-vote consensus with the LLM judge. The judge–consensus κ=0.84 matches the inter-expert Fleiss' κ=0.84, indicating the judge operates at human-level reliability, consistent with the findings of (Zheng et al., 2023a) who report >80% agreement between GPT-4 and human evaluators.

H.3 Individual Annotator vs. LLM Judge

Table 7 reports each annotator's agreement with the judge, showing that the judge aligns comparably with every expert (κ range: 0.73–0.93).

I Detailed Results on RouterWorld-Agentic

I.1 Agentic Benchmark Details

RW-Agentic evaluates routing across three interactive benchmarks: ALFWorld (household tasks), WebShop (web navigation), and WorkArena (enterprise workflows), totaling over 76,000 model–task interactions. The candidate pool comprises six models spanning 14B–235B parameters:

Annotator	n	Agreement	κ
Annotator-S	87	94.3%	0.880
Annotator-A	86	94.2%	0.871
Annotator-D	88	96.6%	0.929
Annotator-H	100	89.0%	0.772
Annotator-P	100	87.0%	0.730
Annotator-W	97	93.8%	0.874

Table 7: Individual annotator agreement with the LLM judge on the 150-sample set.

Model	TTFT (ms)	ITL (ms)	Throughput (tok/s)	Med. Episode Lat. (s)
Ministral-14B	18	3.5	2,764	0.6
Qwen3-Coder-30B	532	7.2	1,126	5.5
Devstral-123B	493	14.6	574	4.9
GPT-OSS-20B	1,137	4.1	987	12.7
GPT-OSS-120B	1,213	6.0	1,169	11.1
Qwen3-235B	2,258	18.8	370	16.4

Table 8: Agentic candidate model pool. TTFT = time to first token; ITL = inter-token latency. Episode latency is the median total time across all tasks.

Ministral-14B, Qwen3-Coder-30B, Devstral-123B, GPT-OSS-20B, GPT-OSS-120B, and Qwen3-235B. Each instance records the user query, the model's step-by-step interaction trace, task success, number of steps, and cumulative episode latency.

Table 8 summarizes per-model statistics. Unlike RW-CORE where latency is dominated by single-turn inference, agentic latency reflects the full episode duration, a product of per-step inference cost and the number of interaction steps required.

I.2 Performance Compression and Policy Collapse

Under the 3-model pool, fully-observable methods cluster tightly with near-zero Acc@1-L variance (σ^2=0.81), compared to σ^2=29.5 in RW-CORE. Expanding to 6 models does not alleviate this—among non-collapsed methods, variance remains low (σ^2=4.35 for Acc@1-L in fully-observable), and the gap to the Oracle upper bound (70.65% Acc@1-L) remains substantial. Under partial observability, the compression worsens further, with multiple routers collapsing into degenerate single-model policies (Table 10).

The root cause lies in the interaction between model planning capability and episode length. Models with stronger planning capabilities resolve tasks in fewer steps (Figure 15): Qwen3-235B requires a median of 18 steps on ALFWorld, while GPT-OSS-20B requires 50. Despite Qwen3-235B having the highest per-step inference latency (16.4s median episode latency), its superior planning yields lower *total* episode cost than weaker models that compensate with many cheap but inefficient steps (Figure 16). This inverts the conventional latency–accuracy trade-off: the largest model is simultaneously the most accurate *and* the most cost-efficient, leaving routers with no meaningful trade-off to exploit.

Without counterfactual feedback, partially-observable routers greedily exploit this dynamic, collapsing into static "Route-To-Best" policies. In the 3-model setting, collapsed routers default to Qwen3-235B; in the 6-model pool, the pattern persists with routers favoring the dominant model. This collapse is rational given the reward structure—the strongest planner minimizes both error and accumulated latency—but it renders routing trivial.

Breaking this plateau requires routing algorithms that reason about multi-step execution dynamics rather than treating agentic tasks as single-turn queries. Capturing non-linear planning–latency interactions demands large-scale rollout environments for exploration-guided training. By providing over 76,000 standardized agentic interactions across 6 models, RW-AGENTIC establishes the largest-scale environment to date for studying these routing bottlenecks in realistic agent deployments.

I.3 Planning Capability and Step Efficiency

Figure 15 shows the median number of interaction steps per model across all three benchmarks. The relationship between model capability and step efficiency is benchmark-dependent: on ALFWorld, GPT-OSS-20B requires a median of 50 steps while most other models cluster around 15–18; on WebShop, Qwen3-235B and Qwen3-Coder-30B are the most step-efficient (median 5 steps); on WorkArena, all models converge to 5–10 steps. These patterns confirm that planning capability—not raw inference speed—is the primary driver of episode cost in agentic settings.

Figure 16 shows the resulting median episode latency. Despite having the highest per-token cost (18.8 ms ITL), Qwen3-235B achieves competitive total episode latency because its superior planning reduces the number of required steps. Conversely, GPT-OSS-20B has low per-token cost (4.1 ms ITL) but incurs the second-highest episode latency due to verbose, multi-step execution.

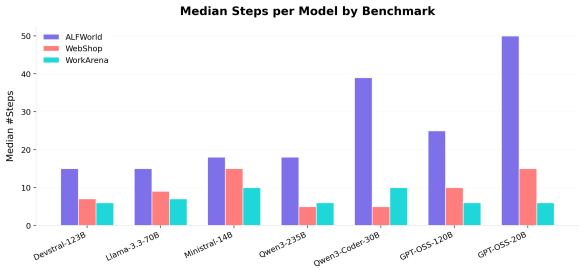

Figure 15: Median interaction steps per model by benchmark. Models with stronger planning capabilities (e.g., Qwen3-235B, Devstral-123B) resolve tasks in fewer steps, particularly on ALFWorld and WebShop.

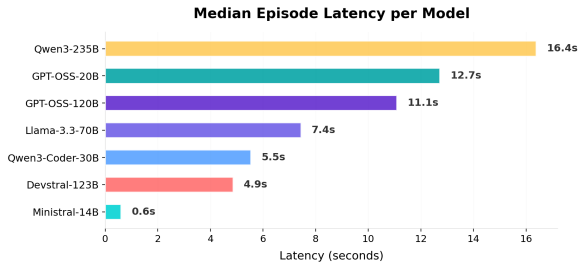

Figure 16: Median episode latency per model. The conventional assumption that larger models are slower does not hold in agentic settings. Planning efficiency can offset higher per-step inference cost.

I.4 Full Routing Results

I.4.1 6-Model Pool

Table 9 presents routing results with the full 6-model candidate pool (Ministral-14B, Qwen3-Coder-30B, Devstral-123B, GPT-OSS-20B, GPT-OSS-120B, Qwen3-235B). BARP achieves the highest Acc@1 under full observability (52.69%), MLP achieves the highest Acc@1-L (39.16%), Naive Bayes achieves the highest DCG (0.4512), and Hybrid-LLM achieves the highest L-DCG (0.4151). Several methods exhibit policy collapse under partial observability: BARP, Avenger, GraphRouter, and Hybrid-LLM all suffer sharp drops in Acc@1-L (below 23%), while methods like Naive Bayes and Emb-FT remain comparatively stable.

I.4.2 3-Model Pool

Table 10 presents results with the original 3-model pool (Qwen3-235B, GPT-OSS-20B, Nemotron-9B). Orange-shaded cells indicate collapsed routing policies that always select a single model; these are excluded from best/second-best comparisons.

J Query Encoder Scaling

We compare `Qwen3-Embedding-0.6B` (1024-dim) against `Qwen3-Embedding-8B` (4096-dim) across four methods that support both encoders. Table 11 reports the results under fully and partially-observable settings.

Scaling the encoder from 0.6B to 8B (13× parameters, 4× embedding dimension) yields surprisingly modest gains. Under full observability, KNN and Avenger improve by less than 1 pp in L-DCG, and MLP gains only 1 pp. BaRP *degrades* sharply, likely because its lightweight decision head (64-dim) cannot effectively leverage the 4096-dim representations, leading to mode collapse.

Under partial observability, the picture is mixed: MLP and BaRP benefit modestly from the larger encoder (+4 pp each), but KNN and Avenger show near-zero improvement. Notably, Avenger with 8B produces identical L-DCG to 0.6B (0.149) under cold-start, which both collapse.

These results suggest that **routing quality is bottlenecked by the training signal, not encoder capacity**. A 13× parameter increase provides at most +1 pp L-DCG under full observability and +4 pp under partial observability, while warm-up initialization (Figure 4, see Appendix B) recovers +15–18 pp with the same 0.6B encoder. Investing in better training data (e.g., diverse warm-up labels) is far more effective than scaling the encoder.

K RouterWorld Prompt Details

This section documents all prompts used in the RouterWorld data collection pipeline and routing methods. Figures 17–24 show the prompts used to roll out candidate models on core evaluation tasks. Figures 25–26 show the agent rollout prompts. Figures 27–32 present the LLM-as-Judge prompts for automated correctness annotation. Figure 33 shows the user query annotation prompt. Figures 34–35 present the generative router prompts used for SFT training and inference.

K.1 Core Task Rollout Prompts

Each benchmark category uses a standardized prompt template to query candidate models during data collection.

K.2 Agentic Task Rollout Prompts

For agentic benchmarks, each candidate model is deployed as an interactive agent producing valid actions in a structured format. Unlike core tasks

Method	Base Model	Fully-Observable				Partially-Observable			
		Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG	Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG
Naive Baselines									
Oracle (Perfect)	–	100.00	100.00	1.0000	1.0000	100.00	100.00	1.0000	1.0000
Oracle (Best-of-N)	–	82.67	70.65	0.7791	0.7521	94.89	77.35	0.8781	0.8313
Random	–	41.64	16.99	0.3010	0.2211	41.64	16.99	0.3010	0.2211
Route-To-Devstral	–	44.41	21.20	0.3519	0.2674	44.41	21.20	0.3519	0.2674
Route-To-Ministral	–	22.93	22.79	0.2286	0.2284	22.93	22.79	0.2286	0.2284
Route-To-OSS-20B	–	39.36	10.77	0.2611	0.1694	39.36	10.77	0.2611	0.1694
Route-To-OSS-120B	–	48.96	14.64	0.3281	0.2181	48.96	14.64	0.3281	0.2181
Route-To-Qwen3-Coder	–	29.56	17.33	0.2441	0.2130	29.56	17.33	0.2441	0.2130
Route-To-Qwen3-235B	–	60.36	13.26	0.3652	0.1967	60.36	13.26	0.3652	0.1967
Representation-based Routers									
KNN	Qwen3-Emb-0.6B	48.20	38.40	0.4414	0.4118	51.59	21.82	0.3774	0.2814
Naive Bayes	Qwen3-Emb-0.6B	51.73	36.12	**0.4512**	0.4021	50.69	**31.15**	**0.4225**	**0.3680**
MLP	Qwen3-Emb-0.6B	45.23	**39.16**	0.4282	0.4122	**58.70**	20.17	0.4014	0.2687
Emb-FT	Qwen3-Emb-0.6B	44.89	36.95	0.4170	0.3925	55.94	23.48	0.4095	0.3045
Hybrid-LLM	DeBERTa Large	46.48	39.02	0.4351	**0.4151**	58.08	13.88	0.3595	0.2049
Graph-Router	Qwen3-Emb-0.6B	46.13	13.47	0.3086	0.2016	53.87	13.40	0.3426	0.2012
BARP	Qwen3-Emb-0.6B	**52.69**	32.73	0.4434	0.3739	22.93	22.79	0.2286	0.2284
Avenger	Qwen3-Emb-0.6B	42.54	36.12	0.3990	0.3848	43.92	11.67	0.2873	0.1790

Table 9: Routing performance on RW-AGENTIC (6-model pool) under fully- and partially-observable settings. **Bold** = best; underline = second best.

```
Solve the following math problem step by step. The last line of your response should only contain your final answer inside a \boxed{} command.
{Question}
Remember to put your answer inside \boxed{}.
```

Figure 17: Rollout prompt for mathematical reasoning benchmarks.

```
Answer the following {subject} question. The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of {candidates}.
{question}
{options}
```

The {subject} field is populated for MMLUPro; other MCQ benchmarks omit it.

Figure 18: Rollout prompt for multiple-choice benchmarks.

where a single prompt–response pair suffices, agentic rollouts involve multi-turn interactions between the model and the environment. Notably, several candidate models required extensive prompt engineering (∼2 hours per model) to reliably extract valid actions in the expected format, as models frequently produced verbose explanations or malformed action strings instead of the required structured output.

K.3 LLM-as-Judge Prompts

After collecting model responses, we use LLM-as-Judge prompts to automatically annotate correctness. Each task category has a dedicated judge prompt that outputs its verdict inside <a>... tags.

K.4 User Query Annotation Prompt

To make the routing dataset realistic, we convert formal benchmark task descriptions into natural user queries. The agent pipeline variant additionally samples from 15 diverse rephrasing styles (e.g., "direct command", "polite request", "search keywords", "Show me...", "Help me pick...") to increase linguistic diversity.

K.5 Generative Router Prompts

The generative routing approach uses the following prompt templates for both SFT training data construction and inference. The model zoo section is dynamically populated with can-

Method	Base Model	Fully-Observable				Partially-Observable			
		Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG	Acc@1 (%)	Acc@1-L (%)	DCG	L-DCG
Naive Baselines									
Oracle (Perfect)	–	100.00	100.00	1.0000	1.0000	100.00	100.00	1.0000	1.0000
Oracle (Best-of-8)	–	86.81	73.46	0.8176	0.7931	88.16	75.19	0.8325	0.8051
Random	–	41.89	26.38	0.3589	0.3161	41.89	26.38	0.3589	0.3161
Route-To-Qwen	–	67.95	38.92	0.5706	0.4971	67.95	38.92	0.5706	0.4971
Route-To-Nemotron	–	43.03	20.54	0.3376	0.2615	43.03	20.54	0.3376	0.2615
Route-To-OSS	–	51.19	40.54	0.4698	0.4462	51.19	40.54	0.4698	0.4462
Generative AI Routers									
Zero-Shot Prompt	Qwen-2.5-0.5B-ins	67.95	38.92	0.5706	0.4971	67.95	38.92	0.5706	0.4971
Gen-SFT	Qwen-2.5-0.5B-ins	76.43	58.32	0.6954	0.6581	75.62	50.59	0.6622	0.6022
Zero-Shot Prompt	Qwen-2.5-1.5B-ins	67.95	38.92	0.5706	0.4971	67.95	38.92	0.5706	0.4971
Gen-SFT	Qwen-2.5-1.5B-ins	76.16	58.32	0.6935	0.6576	74.76	50.05	0.6546	0.5953
Representation-based Routers									
KNN	Qwen3-Embedding-0.6B	75.73	57.84	0.6889	0.6512	72.54	48.76	0.6341	0.5742
MLP	Qwen3-Embedding-0.6B	77.41	59.24	0.7050	0.6673	77.35	55.46	0.6907	0.6411
Emb-FT	Modern-base-0.1B	77.78	58.16	0.7034	0.6604	67.95	38.92	0.5706	0.4971
Emb-FT	Qwen3-Embedding-0.6B	78.05	**60.00**	**0.7118**	**0.6747**	**78.54**	**57.35**	**0.7053**	**0.6583**
Hybrid-LLM	DeBERTa Large	**78.16**	59.35	0.7104	0.6706	78.05	56.16	0.6980	0.6479
Graph-Router	Qwen3-Embedding-0.6B	67.95	38.97	0.5708	0.4974	67.95	38.92	0.5706	0.4971
BARP	Qwen3-Embedding-0.6B	75.89	52.16	0.6696	0.6137	68.22	39.41	0.5741	0.5013
Avenger	Qwen3-Embedding-0.6B	75.08	58.43	0.6877	0.6529	69.84	44.38	0.6021	0.5388

Table 10: Routing performance on RW-AGENTIC under fully- and partially-observable settings. Oracle (Perfect) and Oracle (Best-of-N) denote theoretical and empirical upper bounds, respectively. Orange cells indicate collapsed routing (always selects a single model) and are excluded from best/second-best comparisons. **Bold** = best; underline = second best among non-collapsed methods.

	Fully-Obs			Partially-Obs		
	0.6B	8B	Δ	0.6B	8B	Δ
KNN	.397	.405	+0.8	.222	.223	+0.2
MLP	.430	.439	+1.0	.216	.259	+4.3
BaRP	.298	.162	-13.6	.157	.200	+4.4
Avenger	.371	.379	+0.8	.149	.149	+0.0

Table 11: L-DCG: 0.6B vs. 8B encoder (Δ in pp).

didate model names and their measured latency statistics (TTFT, inter-token latency). The expected output is a single model index wrapped in `<answer>...</answer>` tags.

```
You are an expert Python programmer, and
here is your task:
{question}
Your code should pass these tests:
{tests}
Your code should start with a markdown code
block.
```

Figure 19: Rollout prompt for code generation benchmarks.

```
Answer the logical question, Write a line
of the form "Answer: $ANSWER" at the end of
your response. Let's think step by step.
Here are 3 examples to help you understand
how to answer the question:
{template}
Question: {question}
```

Figure 20: Rollout prompt for BBH logical reasoning (3-shot with chain-of-thought).

```
Your task is to solve a logical reasoning problem. You are given set of statements from which you must logically deduce the identity of a set of characters.

You must infer the identity of each character. First, explain your reasoning. At the end of your answer, you must clearly state the identity of each character by following the format:

CONCLUSION:
(1) ...
(2) ...
(3) ...

### Question:
{question}
```

Figure 21: Rollout prompt for Knights & Knaves deductive reasoning.

```
You are given a *cloze* sentence containing a blank marked by underscores '___'. Only one of the two options correctly fills the blank while preserving commonsense.

After your reasoning, end with a line exactly in the form: 'Answer: $LETTER', where 'LETTER' is A or B.

Sentence: {sentence}
A) {option1}  B) {option2}
```

Figure 22: Rollout prompt for Winogrande commonsense reasoning.

```
Given a conversation history and a current utterance, follow these steps to identify the emotion of the current utterance from the given options. The emotion should be determined based on both the conversation context and the current utterance.
The last line of your response should be of the following format: 'Answer: $LETTER' (without quotes) where LETTER is one of ABCDEFG. Let's think step by step.

{conversation_history}

Current utterance: {utterance}
Options: {options}
```

Figure 23: Rollout prompt for emotion recognition (shared across EmoryNLP, MELD, DailyDialog).

```
Answer the following question.
Question: {question}

Answer:
```

Figure 24: Rollout prompt for open-ended factual QA.

The ALFWorld agent uses ReAct-style few-shot demonstrations. For each of the 6 task types (`put`, `clean`, `heat`, `cool`, `puttwo`, `examine`), 3 in-context examples show the think–act–observe loop. Two prompt variants exist: original and a fixed version with corrected action verbs.

Example: `react_put_0` (abbreviated)

```
You are in the middle of a room. Looking quickly around you, you see a cabinet 4, a cabinet 3, ...
Your task is to: put some spraybottle on toilet.
> think: To solve the task, I need to find and take a spraybottle, then put it on toilet.
OK.
> think: First I need to find a spraybottle. A spraybottle is more likely to appear in cabinet (1-4), countertop (1), ...
OK.
> go to cabinet 2
The cabinet 2 is closed.
> open cabinet 2
You open the cabinet 2. In it, you see a candle 1, and a spraybottle 2.
> take spraybottle 2 from cabinet 2
You pick up the spraybottle 2 from the cabinet 2.
> go to toilet 1
> put spraybottle 2 in/on toilet 1
You put the spraybottle 2 in/on the toilet 1.
```

Figure 25: ALFWorld agent rollout prompt with ReAct few-shot demonstrations.

```
You are an autonomous web agent interacting with a ServiceNow instance.

Your task is to complete knowledge work tasks by interacting with the web browser.

Available actions:
- click(element_id): Click on an element by its ID
- fill(element_id, text): Fill a text field with the given text
- select(element_id, option): Select an option from a dropdown
- scroll(direction): Scroll the page up or down
- goto(url): Navigate to a specific URL
- send_msg_to_user(message): Send a message to the user

Output format: “‘action <your_action_here> “‘

Rules:
1. Output exactly ONE action per response
2. Always use the action code block format
3. Be precise with element IDs from the observation
4. Follow the task instructions carefully
```

Figure 26: WorkArena web agent system prompt for ServiceNow knowledge-work tasks.

```
You are an expert evaluator tasked with judging the correctness of code generation responses.

User Instruction/Problem: [[prompt]]
Model’s Generated Solution: [[response]]

Save your evaluation inside <a></a>, such as <a>correct</a> or <a>incorrect</a>.

Instructions:
1. Extract the code from the model’s response (ignore explanatory text)
2. Evaluate if the code correctly solves the problem
3. Check for logical errors, bugs, or hallucinated functionality
4. Ignore minor stylistic issues if logic is correct
5. Consider the solution correct if it would produce the right output
6. Partially correct solutions that solve the core problem can be considered correct
7. Consider it incorrect if they hardcode
8. Don’t repeat the full code; quick judge whether the code is acceptable
```

Figure 27: LLM-as-Judge prompt for code generation tasks.

```
You are an expert evaluator tasked with judging the correctness of question responses.

Given:
- A ground truth answer: [[ground_truth]]
- A model’s response: [[response]]

Instructions:
1. Extract the final answer (True/False, Yes/No, Entailment/Neutral/Contradiction, etc.)
2. Ignore reasoning; only extract the final answer
3. Normalize both answers (handle case differences, common variations)
4. For semantic equivalence: answers that mean the same thing should be correct
5. Respond with exactly one word: “correct” or “incorrect”
```

Figure 28: LLM-as-Judge prompt for open-ended QA tasks.

```
You are an expert evaluator tasked with judging the correctness of multiple-choice question responses.

Given:
- A ground truth answer: [[ground_truth]]
- A model’s response: [[response]]

Instructions:
1. Extract the final answer (look for patterns like A, B, C, D, E, F)
2. Some answers may contain reasoning. Ignore it; only extract the final answer
3. Compare it with the ground truth answer
4. Respond with exactly one word: “correct” or “incorrect”
```

Figure 29: LLM-as-Judge prompt for multiple-choice tasks.

```
You are an expert evaluator tasked with judging the correctness of SQL query responses.

User Instruction: [[prompt]]

Given:
- A ground truth SQL query: [[ground_truth]]
- A model's generated SQL query: [[response]]

Instructions:
1. Extract the SQL query from the model's response (ignore explanatory text)
2. Ensure the table and column names are not hallucinated
3. You can ignore DISTINCT, or ROUND such details
4. Compare the logical equivalence of both queries
5. Consider queries correct if semantically equivalent, even if syntactically different
6. Partially correct can also be considered as correct
```

Figure 30: LLM-as-Judge prompt for SQL generation tasks (relaxed variant).

```
You are an expert evaluator tasked with judging the correctness of tool use and function calling responses.

Given:
- A ground truth tool call: [[ground_truth]]
- A model's response: [[response]]

Instructions:
1. Extract the tool/function call (look for function names, parameters, arguments)
2. Verify: correct tool name, all required parameters present, values match, types correct
3. Ignore reasoning text—focus only on the actual tool call
4. Minor formatting differences are acceptable if semantic meaning is identical
5. Count it correct if they use function name such as "Currency Finder" not "currency.find"
6. Don't have to be over-strict (not case-sensitive, not bool-sensitive)
```

Figure 31: LLM-as-Judge prompt for tool use / function calling tasks.

```
You are an expert evaluator tasked with judging the quality and appropriateness of chatbot responses in personal conversations.

Given:
- A conversation context: [[prompt]]
- A model's response: [[response]]

Instructions:
1. Evaluate if the response is contextually relevant
2. Check if the tone is suitable for personal/casual conversation
3. Assess if the response maintains appropriate boundaries
4. Verify no harmful, offensive, or inappropriate content
5. Consider if the response moves the conversation forward constructively

"good" if: addresses context naturally, suitable tone, helpful, maintains boundaries.
"bad" if: ignores context, overly formal/robotic, unhelpful, inappropriate content.

Don't be too strict since NL allows for rooms. Only brief reasons (one sentence) then evaluation.
```

Uses a "good/bad" scale instead of "correct/incorrect".

Figure 32: LLM-as-Judge prompt for personal chat tasks.

```
You are simulating a human USER interacting with an AI assistant. Generate a natural, colloquial query that a real user would ask for the given task.

# Input
You will receive a formal task prompt that may include:
- User requests and requirements
- Referenced data or context
- Success criteria or expected outcomes

# Guidelines
Transform the formal prompt into natural, colloquial language by:
- Using casual phrasing (as a real user would speak)
- Omitting references to "tabular data" or backend systems
- For objective answer tasks (multiple choice, yes/no):
- Avoid formal answer formats
- Integrate answer choices naturally into the question
- Removing any meta-instructions or formatting requirements
- Keeping the core intent and requirements clear
- Don't always start with "Hey, I'm ...", natural and concise please

# Output Format
Return ONLY the converted user query wrapped in tags: <a>[Your natural query here]</a>

# Complete Task Prompt
[[prompt]]
```

Figure 33: User query annotation prompt. Converts formal benchmark descriptions into natural conversational-form queries for the routing dataset.

```
You are an expert LLM router. Given a user query, you need to select the best model that can:
1. Answer the query correctly
2. Respond as fast as possible

Available Models:
Index 0: Mistral-7B-Instruct-v0.3 (TTFT: 100.0ms, ITL: 0.46ms)
Index 1: Qwen3-235B-A22B-Instruct-2507 (TTFT: 2258.0ms, ITL: 2.70ms)
...remaining models...

Given the following query, predict which model is most likely to answer correctly AND be the fastest among correct models.

Query: {query}

Instructions:
- Consider model capabilities and latency
- Output ONLY the model index number (0-{max_model_idx}) that you predict will be correct and fastest
- Format your answer as: <answer>MODEL_INDEX</answer>

Your prediction:
```

Training Target

<answer>{label}</answer> where label = index of the fastest correct model

Figure 34: Generative router prompt used for SFT training and inference. The model zoo section is populated dynamically with measured latency statistics.

```
You are an expert LLM router. Given a user query, select the best model that answers correctly and fastest.

{model_zoo}

Here are similar queries from past routing decisions (correct model shown):
Example 1 [MATH500]: "Solve for x in 3x+7=22" → Model 4 (deepseek-math, 342ms)
Example 2 [MMLUPro]: "Which element has atomic number 79?" → Model 0 (Mistral-7B, 187ms)
...

Now route this new query:
Query: {query}

Based on the similar examples above, output ONLY the model index (0-{max_model_idx}) that will answer correctly and fastest.
Answer: <answer>
```

Figure 35: RAG-augmented router prompt. Retrieved similar queries from the training set are included as few-shot examples to improve routing accuracy.