COMPASS: ROUTING VIA CAPABILITY-ORIENTED MODEL PROFILING WITH ADAPTIVE SEARCH, SELECTION, AND TOOL USE

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM routing aims to select a capable model at low cost, but specialized routers can require updates as models and tasks change. We propose **COMPASS**, an agentic router in which an LLM follows a routing recipe, gathers performance and cost evidence through tools, and optionally inspects candidate answers before selecting a model. Our evaluations on LLMRouterBench and DeepSWE show that agentic routing can improve accuracy and lower model-call cost relative to specialized and trained routing baselines, without routing-specific training. On LLMRouterBench, COMPASS reaches 69.85% accuracy at $139.57, compared with RLCascadeRouter's 69.14% at $180.22. On DeepSWE under fixed five-fold cross-validation, it reaches 78.1% at $215, compared with 74.1% at $500 for the best single model. COMPASS also supports routing across benchmarks and adapts to an expanded model pool by reading new evaluation records, without changing the router's weights or prompt.

1 INTRODUCTION

The ecosystem of large language models has fragmented into distinct capability–price tiers. Frontier models dominate benchmarks but are expensive and often slow; small open-weight models are cheap and fast, and for many task distributions they are entirely sufficient. Consider a user requesting a simple Python script: this task does not require a frontier model, and a cost-effective model would complete it at a fraction of the price. The difficulty is that this knowledge is available only *in hindsight*: one would need to query both the cheap and the frontier model to know whether the cheap model actually sufficed. The routing problem is precisely the problem of estimating this counterfactual *before* committing to a model.

Routing must navigate a three-way trade-off:

- **Cost.** Cheaper models reduce per-token spend, but only if they succeed; failed attempts followed by escalation can cost more than routing correctly the first time.
- **Latency.** A cheap model with low throughput that emits many tokens can take far longer end-to-end than a stronger model that solves the task in substantially fewer tokens. Cost-optimal and latency-optimal choices frequently diverge.
- **Accuracy.** Some users simply want the strongest available response regardless of price.

Most prior work optimizes a single axis, usually cost reduction at near-constant quality, because cost is the easiest axis to measure and experiment on (Chen et al., 2024a; Ong et al., 2025). Optimizing for accuracy instead pushes evaluation toward frontier models, where per-experiment API costs are substantial and public fine-grained evaluation data is scarce.

Routing recipes. Users can specify what they value without estimating how difficult a query will be. A performance-focused recipe asks the router to prioritize reliable completion; a cost-optimized recipe asks it to seek cheaper configurations when the evidence supports comparable quality. Qualitative levels such as low, medium, or high can express the strength of these preferences. The recipe provides guidelines for gathering evidence and making a decision, while the router assesses the

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

query using that evidence. Numerical trade-off parameters can still be used inside tools and experimental sweeps; users need not tune them for each request.

The staleness problem. When model-specific performance estimates are encoded in a trained router, changes in the model pool or task distribution can require new supervision or updates. Methods that support unseen models reduce this burden, but still need useful evidence about their capabilities. We instead keep evaluation outcomes in an external corpus that the router can query at inference time. New models can be added by collecting and inserting their evaluation records, without updating the routing agent's weights. This shifts the maintenance burden to evidence collection, whose cost and coverage remain important limitations.

Our proposal. We separate *routing knowledge* from *routing skill*. Knowledge about which model scored what, at what cost, and on which task lives in an external evaluation corpus. The skill to query that corpus and decide under a routing recipe comes from the tool-use capabilities of an existing LLM. COMPASS requires no routing-specific weight training: the agent follows the recipe, adaptively explores evidence, and submits a result when it judges the evidence sufficient. This yields three design requirements:

1. **Recalibration through evidence.** New models and benchmarks are incorporated by inserting evaluation records into the corpus; the router's weights are untouched.
2. **Recipe-oriented routing.** The router follows input guidelines, including performance-focused and cost-optimized recipes with qualitative preference levels.
3. **Low routing overhead.** Evidence acquisition and agent inference must be economical enough to preserve any savings in model execution.

Contributions.

1. We formulate routing as evidence gathering through tools, separating updatable model records from an existing LLM's decision-making capabilities (Section 3).
2. We implement COMPASS, which follows routing recipes to gather evidence, inspect candidate answers, and select a model or stop unresolved (Section 4).
3. We evaluate COMPASS against routing baselines on LLMRouterBench and under five-fold cross-validation on DeepSWE. COMPASS reaches 78.1% accuracy on DeepSWE and 76.7% using only TerminalBench evidence. Expanding the model pool improves accuracy and lowers model-call spend without retraining; ablations measure the effects of router choice and model names (Section 5).

2 Related Work

Trainable model routing. One-step routers learn to select a model or execution configuration from preferences and difficulty estimates (Ong et al., 2025; Ding et al., 2024), predicted performance (Xue et al., 2026; Fernandez et al., 2026; Liu et al., 2024b; Zhuang et al., 2024), query–model representations (Chen et al., 2024b; Feng et al., 2024; Jitkrittum et al., 2026), online feedback (Panda et al., 2025), or latent expert interactions (Fein-Ashley et al., 2025). They commit before inspecting a completed answer and therefore cannot revise that query's route using post-generation evidence. Sequential systems can instead purchase more precise estimates (Fisch et al., 2026), inspect an answer before stopping or escalating (Chen et al., 2024a; Shah & Shridhar, 2025; Chuang et al., 2024; Dekoninck et al., 2025; Gupta et al., 2024; Aggarwal et al., 2024), or learn multi-call orchestration (Nielsen et al., 2026; Shao et al., 2025). Across both families, trained predictors, scorers, and policies may need refreshed outcomes or updates as model pools and task distributions change. Some methods explicitly support new models, but the reliability of transfer across pools remains an empirical question.

Training-free model routing. Training-free routers reuse stored query–model outcomes without fitting a dedicated neural policy. The Avengers, Avengers-Pro, and UniRoute's k-means variant summarize model performance in prompt clusters (Zhang et al., 2025b;a; Jitkrittum et al., 2026).

Nearest-neighbor variants in RouterBench and CARROT, together with benchmark-based and online kNN routers, transfer observed success or inference-cost estimates from similar evaluated queries (Hu et al., 2024; Shnitzer et al., 2023; Somerstep et al., 2025; Li, 2025; Wu & Silwal, 2025). Eagle blends global and local Elo rankings (Zhao et al., 2024), while ProxRouter reweights nearby evidence to improve outlier robustness (Patel et al., 2025). The quality of these estimates depends on the coverage and relevance of stored examples: when neighbors are sparse, noisy, or close in embedding space yet different in difficulty or required capability, local model rankings can be unstable or misleading. Global rankings and proximity weights can reduce this risk, but a weakly supported query may still require other evidence.

These approaches expose two sources of fragility: trainable predictors and policies may require new supervision or retraining as model pools change, and their transfer to unseen pools remains uncertain; neighbor-based training-free estimates can become unreliable when relevant examples are sparse, weakly related, or noisy. COMPASS separates model-specific knowledge from routing skill: outcomes and inference costs reside in an updatable evaluation corpus, while an existing LLM agent uses anonymized model identities to decide which evidence to acquire. Adding a model requires calibration outcomes rather than changes to the agent's weights. Neighbor retrieval is only one optional evidence tool; the agent can also consult model profiles and quality–cost estimates, inspect a generated answer, request further evidence, or stop unresolved. More informative tools can enrich this search as they become available, provided their inference cost and effect on final-answer reliability are assessed.

3 METHOD

Routing Problem. Let $(X, Y) \sim P$ denote a task and its success criterion. The finite pool $\mathcal{M}$ is fixed across queries within a deployment and throughout each routing episode, but may differ from the pool used to develop the router. Each $m \in \mathcal{M}$ specifies a model and its execution settings, including prompt, decoding, tools, and reasoning budget. Executing m yields a possibly random answer $\widehat{Y}_m$ at realized inference cost C_m. With binary failure indicator L_m, define

$$\begin{aligned} s(x, m) &:= \Pr(L_m = 0 \mid X = x), \\ \kappa(x, m) &:= \mathbb{E}[C_m \mid X = x]. \end{aligned} \tag{1}$$

For a system-defined completion target $\rho \in (0, 1]$, the ideal single-call router selects

$$r^*_\rho(x, \mathcal{M}) \in \operatorname*{arg\,min}_{m \in \mathcal{M}:\, s(x,m) \geq \rho} \kappa(x, m). \tag{2}$$

This is a completion-first objective: among configurations meeting the target, it chooses the cheapest in expected inference cost, without a user-specified spending cap. If no configuration qualifies, a single preselected execution cannot meet the target. The true $s(x, m)$ and $\kappa(x, m)$ are unknown to the router.

Sequential Evidence Acquisition. The agentic router can inspect evidence and candidate answers before returning one. After t interactions, its history is $h_t = (x, \mathcal{M}, (a_1, o_1), \ldots, (a_t, o_t))$, where a_i is an action and o_i its observation. If K_t answers have been previewed, the router policy π_r chooses from

$$\begin{aligned} \mathcal{A}(h_t) = \mathcal{A}_{\text{ev}}(h_t) &\cup \{\text{preview}(m) : m \in \mathcal{M}\} \\ &\cup \{\text{accept}(j) : 1 \leq j \leq K_t\} \cup \{\text{unresolved}\}. \end{aligned} \tag{3}$$

Evidence actions may retrieve outcomes from similar tasks, query model profiles or predictive routers, and assess a previewed answer. A preview runs configuration m and stores its answer for possible acceptance; accepting it does not run m again. Tool availability and call limits restrict $\mathcal{A}(h_t)$.

Objective and Stopping. Let τ be the terminal interaction, and let c_i be its incremental inference cost, including router reasoning, evidence calls, and previews. Let Z_r indicate whether the returned answer completes the task, with $Z_r = 0$ for an unresolved outcome. We seek a finite-terminating

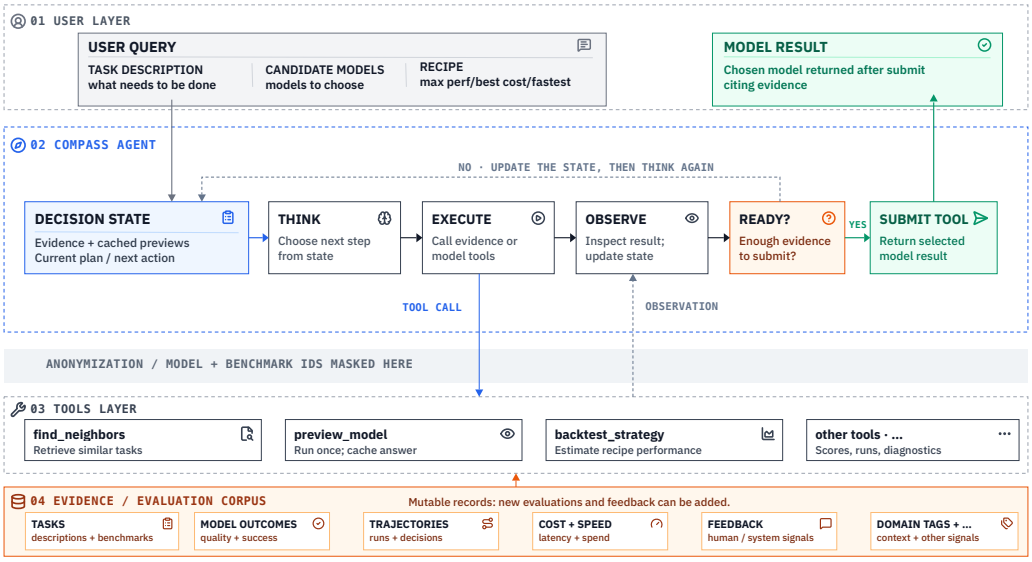


Figure 1: COMPASS queries evaluation records, optionally pays to preview an answer, and selects a model and reasoning-effort level. New model records can be added without retraining the router.

policy that solves

$$\begin{aligned} \min_{\pi_r} \quad & \mathbb{E}_{\pi_r}\left[\sum_{i=1}^{\tau} c_i\right] \\ \text{s.t.} \quad & \Pr_{\pi_r}(Z_r = 1 \mid X = x) \geq \rho. \end{aligned} \tag{4}$$

The constraint applies for P_X-almost every x and concerns the final answer, not the marginal success rate of the model that generated it. The cost counts all calls, including evidence queries and rejected previews.

At history h_t, the ideal remaining-cost comparison for an action a is

$$W(a \mid h_t) := \inf_{\pi \in \Pi_\rho(h_t, a)} \mathbb{E}^{\pi}\left[\sum_{i=t+1}^{\tau} c_i \,\middle|\, h_t\right], \tag{5}$$

where $\Pi_\rho(h_t, a)$ contains admissible policies starting with a that meet the completion target conditional on h_t; $W = +\infty$ if none exists. Past costs are sunk in this comparison. The router need not compute W or know the true success probabilities: it uses the available evidence to decide whether to accept an answer, gather more evidence, preview another model, or stop unresolved. Its confidence estimates may be implicit or uncalibrated, so the completion target must be evaluated using actual final outcomes.

4 COMPASS

COMPASS gives an existing LLM a routing prompt (a *recipe*), evaluation corpus $\mathcal{E}$, candidate list, and accuracy–cost preference $\alpha \in [0, 1]$. For each query, it gathers evidence and submits a configuration (m, e) (a model alias and reasoning-effort level) with free-text evidence and a self-reported, uncalibrated confidence label. Different effort levels are separate choices with measured pass rates and costs. No routing-specific weights are trained.

4.1 Evidence and tools

A per-run access mask fixes the visible tasks, candidates, and aliases. The corpus stores each attempt's outcome, USD cost, token usage, and available output (patch, transcript, or answer). DeepSWE has roughly four attempts per cell; LLMRouterBench has one. A preview therefore exposes

Table 1: COMPASS tools. Evidence reads are free; answer previews add model-call cost.

Tool	Input/output	Purpose
`find_similar_tasks`	$q \mapsto \{t_i\}$	Nearest visible tasks
`get_task_description`	$t \mapsto$ text	Task text shown to models
`lookup_scores`	$(\{t\}, \{m\}) \mapsto \{(s, c)\}$	Outcomes and costs; optional grouping
`list_configurations`	$\emptyset \mapsto \{(m, e)\}$	Allowed candidates
`neighbourhood_evidence`	$q \mapsto$ table	All candidates' neighbour outcomes and benchmark composition
`reliability_table`	$\emptyset \mapsto$ table	Overall and ecosystem pass rates, costs, frontier
`statistic_recommendation`	$\emptyset \mapsto (m, e)$	kNN choice at α
`benchmark_diagnostics`	$\emptyset \mapsto$ (gain, AUC)	Routing gain over best constant; judge accuracy
`backtest_policy`	$\{\pi\} \mapsto \{(\bar{s}, c, \Delta, \text{CI})\}$	Held-out policy scores and paired comparisons
`preview_answer`	$(m, e) \mapsto \tau$	Paid stored attempt on current task
`submit_recommendation`	$(m, e,$ evidence, conf.$)$	End session

one sample on DeepSWE but the sole scored answer on LLMRouterBench. Task descriptions are embedded once for nearest-neighbour retrieval, restricted to visible tasks.

For each mask, we precompute configuration pass rates and costs, ecosystem-specific rates, the accuracy–cost frontier, and recommendations over the k nearest tasks. Diagnostics estimate task-specific routing gains and how well an output judge predicts scores. Policy comparisons hold out each visible task in turn and report paired confidence intervals against the cheapest policy. Table 1 lists the interface.

Tools have individual call limits; identical calls are deduplicated, and recipes allow 4–16 calls total. Reading corpus evidence adds no model-call cost. Each preview is charged even if rejected. Precomputed tables reduced calls from six to two or three per task while maintaining or improving accuracy: the agent compares evidence, decides whether to preview, and stops instead of rebuilding statistics.

4.2 Routing recipes

Recipes specify the objective, tool order, and selection rule. We evaluate four families:

- **Single pick:** read reliability or neighbourhood evidence, choose from its accuracy–cost frontier using α, and submit; one or two calls, no previews.
- **Cascade:** start with the kNN recommendation, inspect its answer, and accept or try the next candidate (Section 3).
- **Conditional preview:** allow all evidence sources, but enable previews only under preset conditions, such as tied leaders, conflicting evidence, or benchmark diagnostics.
- **Choose a policy:** compare predefined policies with `backtest_policy`, then apply one. The agent cannot invent policies.

Shared prompt segments keep selection rules identical across arms. Paraphrasing the single-pick rule changed DeepSWE pass@1 by 5.5 points; replacing its informal trade-off with an executable objective changed it by 9.5 points.

4.3 Model identities and updates

Aliases are generated per run and resolved outside the agent. They hide names and reputation, although scores, prices, or output style may reveal identity (Section 5.7.1). Adding candidates requires only their records and a candidate-list update, with the router's weights unchanged (Section 5.6).

4.4 Execution and evaluation

Sessions end on submission or budget exhaustion; exhausted sessions are unresolved and retried only under the transient-failure policy. Task cost includes every previewed configuration and the submitted configuration at their measured per-attempt costs; router inference is reported separately. Corpus-only recipes score the chosen configuration's mean pass rate; cascades score the inspected, accepted attempt. We report these scoring conventions separately (Section 5).

Evidence and evaluation tasks are disjoint. Visible records supply COMPASS's evidence and baseline training data, never router weight updates. Tables are rebuilt per fold; removing held-out tasks must leave them byte-identical. We study one routing decision per task, not model switching within a conversation.

5 Experiments

5.1 Evaluation data

Routing cannot be evaluated from leaderboard aggregates. A router makes a decision per task, so evaluating it requires, for every task and candidate model, whether the model solved the task and how much the attempt cost. This per-task record is the scarce resource and determines which benchmarks we can use.

Our main routing evaluations use three sources. **LLMRouterBench** (Li et al., 2026) provides pre-collected per-prompt, per-model results for thirteen models across ten benchmarks: AIME (Mathematical Association of America, n.d.), ArenaHard (Li et al., 2024), GPQA (Rein et al., 2023), Humanity's Last Exam (HLE) (Phan et al., 2025), LiveCodeBench (Jain et al., 2024), LiveMathBench (Liu et al., 2024a), MMLU-Pro (Wang et al., 2024), SimpleQA (Wei et al., 2024), SWE-bench (Jimenez et al., 2023), and τ^2-Bench (TAU2) (Barres et al., 2025). We use its released record dump with the performance–cost configuration's filters. It is also the corpus used for our head-to-head baseline comparison. Its main limitation is that it contains *one record per prompt and model*: there are no repeated trials, so pass@k and per-task variance cannot be measured.

DeepSWE (Huang et al., 2026) and **TerminalBench** (Merrill et al., 2026) publish traces. Their repeated trials per cell (four on DeepSWE) make the pass@k and ceiling analysis in §5.5 possible, since we can split trials into disjoint evidence and evaluation sets. The traces also let the router inspect what a model did rather than only whether it passed. Our DeepSWE catalog contains 31 617 trials over 113 tasks and 28 model/effort configurations. The paired TerminalBench cohort contains 66 tasks on an exact 18-configuration roster.

For the ceiling analysis in Section 5.5, we also use repeated-trial records from **FrontierSWE** (Kondra et al., 2026).

5.2 LLMRouterBench: head-to-head on a 70/30 split

We reproduce the LLMRouterBench split with seed 42, using 70% of tasks for training and 30% for testing. The three lowest-cost COMPASS points in Figure 2 use pilot measurements with a $+0.5$ percentage-point display adjustment and costs scaled to the test-set size.

5.2.1 Baselines

We run FrugalGPT, GraphRouter, and Avengers-Pro on this split using stored records and upstream implementations where available. We report our measured results for these baselines and include one published RLCascadeRouter operating point for comparison. FrugalGPT uses thirteen per-model DistilBERT answer scorers and a shared scorer, with cascade orders and thresholds searched over 1885 ordered cascades. GraphRouter uses the released GNN configuration with seed 1881 and 2000 epochs. Avengers-Pro uses k-means over the 8558 training embeddings and top-k cluster routing.

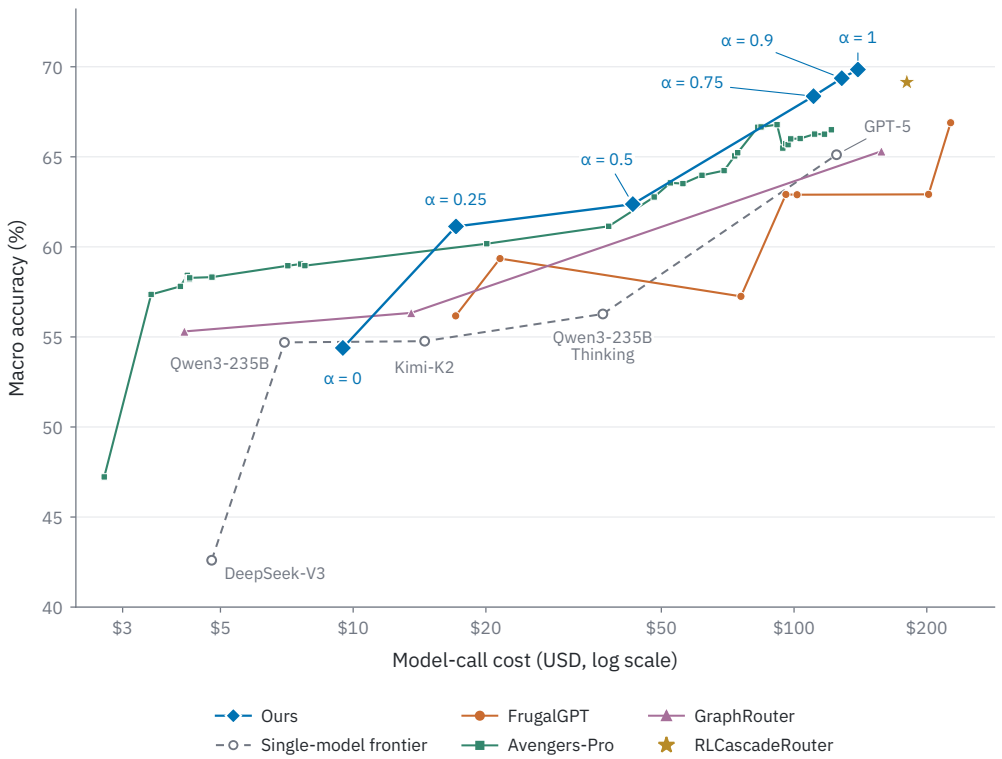


Figure 2: Accuracy and model-call cost of COMPASS, single models, and routing baselines on LLMRouterBench.

5.2.2 Results

Figure 2 shows how COMPASS trades model-call cost for accuracy. Its most accurate setting reaches **69.85% at $139.57**, compared with the plotted RLCascadeRouter reference at 69.14% and $180.22. COMPASS also reaches 69.36% at $128.30, exceeding GPT-5's 65.12% at a similar cost of $124.82.

At a lower cost, COMPASS reaches 68.37% at $110.69. The best recorded Avengers-Pro setting reaches 66.79% at $91.50, while FrugalGPT reaches 66.89% at $226.54 and GraphRouter reaches 65.31% at $157.79. COMPASS therefore provides the highest accuracy in the figure, while Avengers-Pro remains competitive at lower budgets. These costs exclude router inference.

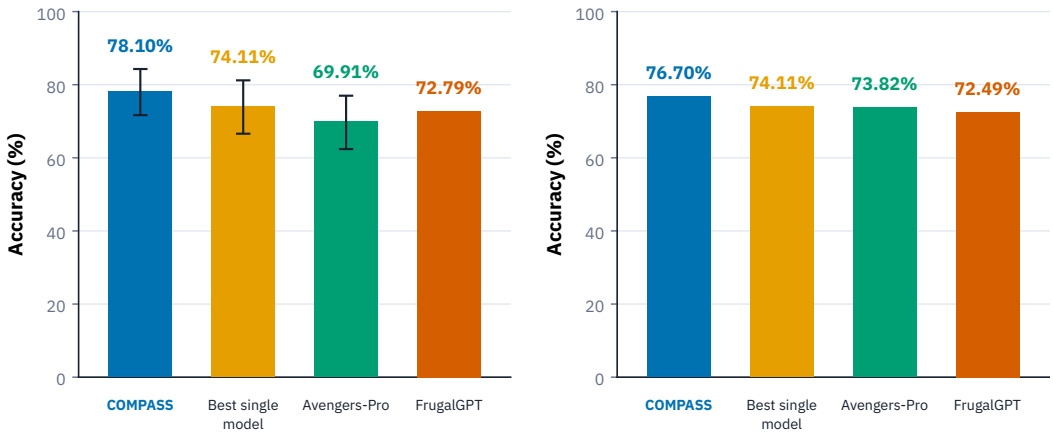
5.3 DeepSWE: five-fold cross-validation

DeepSWE has only 113 tasks, so a 70/30 split leaves a small test set and produces highly variable results. We therefore use five-fold cross-validation, holding each evaluation task out of the routing evidence and keeping COMPASS's policy fixed across folds. Accuracy averages the recorded trial outcomes: one success in four trials contributes 25%.

Figure 3a shows COMPASS at **78.1%**, compared with 74.1% for the best single model (`gpt-6-astra/xhigh`), 69.9% for Avengers-Pro, and 72.8% for FrugalGPT. COMPASS uses $215 in model-call spend, compared with $500 for the best single model. The accuracy estimates favor COMPASS, although the confidence intervals remain wide on this small task set.

5.4 Out-of-distribution routing

Here the router receives evidence from *one benchmark only* and must route tasks from a different benchmark. We use TerminalBench evidence to route 113 DeepSWE tasks, with three routing repeats and the same 18 model/effort configurations in both cohorts.



(a) DeepSWE

(b) TerminalBench → DeepSWE

Figure 3: DeepSWE accuracy with in-domain evidence (left) and TerminalBench evidence (right). Error bars show available 95% task-bootstrap intervals.

With TerminalBench evidence, COMPASS reaches 76.7% accuracy on DeepSWE, compared with 74.1% for the best single model, 73.8% for Avengers-Pro, and 72.5% for FrugalGPT (Figure 3b). The paired confidence interval for COMPASS's advantage over the best single model includes zero. The result shows that evidence from another benchmark can support useful routing, but does not establish an accuracy improvement over that model.

5.5 How much can routing improve accuracy?

Routing helps only when different models are reliably better on different tasks. To estimate this opportunity, we use repeated trials on DeepSWE and FrontierSWE: one group of trials selects a model, and a disjoint group evaluates it. Across 60 random splits, we compare selection using evidence from the target task itself, the most behaviorally similar other task, and a random task against always using the best single model.

On DeepSWE, evidence from the task itself yields a 9.0-point gain, while evidence from the most similar other task yields 4.0 points (Figure 4 in Appendix A). This suggests useful routing opportunities, but only part of that opportunity is accessible through related tasks. On FrontierSWE, neither approach beats the best single model, leaving little measured benefit from per-task selection.

The point is to distinguish apparent oracle gains from gains a router can actually recover. Choosing and scoring a model on the same outcomes exaggerates the opportunity by selecting lucky trials. LLMRouterBench has only one record per task and model, so it cannot support this disjoint-trial check; its oracle gap alone is not evidence of achievable routing gains.

5.6 Generalization to a new model pool

We expand the DeepSWE catalog from 18 to 28 models by adding their evaluation records, then rerun COMPASS on the same 113 tasks under fixed five-fold cross-validation. With `gpt-5.6-luna/medium` as the router, accuracy rises from 63.9% to **78.1%**, while model-call spend falls from $474 to **$215**. The routing model, prompt, and policy remain fixed: COMPASS adopts the new models directly from their records, without retraining.

5.7 Ablations

5.7.1 Does anonymity matter?

We compare opaque aliases with real model names while keeping the router, prompt, candidate pool, and tasks fixed. On LLMRouterBench, revealing names changes macro accuracy by -0.49 points

for `gpt-5.6-luna` and $+0.47$ points for DeepSeek-V4.1-Flash. These single-run comparisons show small, opposite changes, with no consistent accuracy benefit from revealing model identities.

Names nevertheless affect the choices: Luna changes 19% of its selections, compared with 6% for DeepSeek. Both shift slightly toward GPT-5, whose selection share rises from 35.7% to 37.8% for Luna and from 33.6% to 34.1% for DeepSeek. This suggests that model-specific priors can influence routing even when aggregate accuracy barely changes. Anonymization removes explicit model names as a cue for memorized reputation or flagship preference, encouraging decisions based on the supplied evidence.

5.7.2 How much does the routing model matter?

We swap the routing LLM while keeping the agent, prompt, objective, candidate pool, and tasks fixed. On the LLMRouterBench ablation, `gpt-5.6-luna`, `gpt-5.6-sol`, and DeepSeek-V4.1-Flash differ by only 1.15 macro accuracy points. DeepSeek selects models with 22% less model-call spend than Luna, at a cost of 0.86 accuracy points. Changing the router therefore affects spending more than accuracy in this setting; these costs exclude the router's own inference bill.

Because these LLMs have not been trained for our routing objective, they can interpret the same cost–accuracy preference differently. DeepSeek tends to favor cheaper configurations, which can sacrifice accuracy when the model pool offers a wider range of choices. On DeepSWE with fixed five-fold CV, it reaches 68.4% at \$105, compared with Luna's 78.1% at \$215. Router choice thus matters more in this setting: lower model-call spend can come with a substantial accuracy loss.

6 Limitations

- **Existing LLMs as routing agents.** COMPASS uses an existing LLM; we have not trained a specialized model for routing. The results motivate investigating such training as a next step, particularly to improve evidence use and reduce inference overhead, but do not establish that it will improve performance.
- **Evaluation cost and coverage.** Collecting per-task outcomes and trajectories across models is expensive. Routing quality depends on the coverage, reliability, and relevance of those records. Adding a model requires new evidence even though it does not require retraining the agent, and sparse or mismatched data can lead to poor choices.
- **Distribution-dependent routing.** Different task distributions have different routing profiles, and the same router does not perform best across all datasets. Strong results on one cohort do not establish reliable transfer to another.
- **Routing overhead and uncertainty.** Agent inference can consume the savings from cheaper model choices. DeepSWE estimates have wide confidence intervals, and most experiments use one routing realization per task. The reported task-bootstrap intervals therefore do not capture all routing stochasticity.

7 Conclusion

COMPASS uses an existing LLM to follow routing recipes and adaptively gather evidence from an updatable evaluation corpus. This separation lets new models be incorporated without retraining the routing agent. Experiments on LLMRouterBench and DeepSWE demonstrate useful accuracy–cost tradeoffs, while the routing-cost analysis and cross-benchmark transfer results show that gains depend on both the evidence and the task distribution.

A next step is cache-aware multi-turn routing, in which the router revisits its decision as a session reveals more about the task. Switching models can incur fresh context-processing costs and lose caching benefits, so a useful policy must weigh these costs against the expected benefit of continuing, switching, or delegating a subtask. Evaluating such policies requires session trajectories with recorded switching decisions, outcomes, and costs. Constructing and evaluating this data is future work; the present experiments address individual tasks.

AI USE STATEMENT

We used generative AI tools to aid and polish writing, including drafting, editing, and LaTeX formatting; for retrieval and discovery of relevant literature; and for research execution, including software development and faster experimentation. These activities were carried out under human oversight. The authors reviewed and revised AI-assisted text, checked cited work against original sources, and verified AI-assisted code, experimental outputs, and analyses. We take responsibility for the final content of this work, including its claims, results, and artifacts.

REPRODUCIBILITY STATEMENT

The routing objective and action space are specified in Section 3, and the COMPASS architecture and tool interface in Section 4. Section 5 describes the evaluation corpora, the seed-42 split, baseline implementations, and cross-validation and transfer protocols. COMPASS uses existing LLMs without routing-specific weight training. Baseline configurations and the single RLCascadeRouter reference point are documented in Section 5.2.1; routing overhead and statistical limitations are discussed alongside the results and in Section 6.

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 Mausam. AutoMix: Automatically mixing language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 131000–131034. Curran Associates, Inc., 2024. doi: 10.52202/079017-4164. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/ecda225cb187b40ea8edc1f46b03ffda-Paper-Conference.pdf.

Victor Barres, Honghua Dong, Soham Ray, Xujie Si, and Karthik Narasimhan. τ^2-Bench: Evaluating Conversational Agents in a Dual-Control Environment, 2025. URL https://arxiv.org/abs/2506.07982.

Lingjiao Chen, Matei Zaharia, and James Zou. Frugalgpt: How to use large language models while reducing cost and improving performance. *Transactions on Machine Learning Research*, 2024a. URL https://openreview.net/forum?id=cSimKw5p6R.

Shuhao Chen, Weisen Jiang, Baijiong Lin, James T. Kwok, and Yu Zhang. RouterDC: Query-Based Router by Dual Contrastive Learning for Assembling Large Language Models, 2024b. URL https://arxiv.org/abs/2409.19886.

Yu-Neng Chuang, Prathusha Kameswara Sarma, Parikshit Gopalan, John Boccio, Sara Bolouki, Xia Hu, and Helen Zhou. Learning to Route LLMs with Confidence Tokens, 2024. URL https://arxiv.org/abs/2410.13284.

Jasper Dekoninck, Maximilian Baader, and Martin Vechev. A unified approach to routing and cascading for LLMs. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 12987–13010, 2025. URL https://proceedings.mlr.press/v267/dekoninck25a.html.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruhle, Laks V. S. Lakshmanan, and Ahmed Hassan Awadallah. Hybrid LLM: Cost-Efficient and Quality-Aware Query Routing, 2024. URL https://arxiv.org/abs/2404.14618.

Jacob Fein-Ashley, Dhruv Parikh, Rajgopal Kannan, and Viktor Prasanna. Mixture of Thoughts: Learning to Aggregate What Experts Think, Not Just What They Say, 2025. URL https://arxiv.org/abs/2509.21164.

Tao Feng, Yanzhen Shen, and Jiaxuan You. GraphRouter: A Graph-based Router for LLM Selections, 2024. URL https://arxiv.org/abs/2410.03834.

Nigel Steven Fernandez, Branislav Kveton, Ryan Rossi, Andrew Lan, and Jack Wang. RADAR: Reasoning-ability and difficulty-aware routing for reasoning LLMs. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 109429–109457, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/b1f7288854d3bd476c17725c2d85967f-Paper-Conference.pdf.

Adam Fisch, Shubhendu Trivedi, Fantine Huot, William W. Cohen, Michael Kaisers, Mirella Lapata, Kate Larson, and Jacob Eisenstein. Pandora's AI Model Routing Box: Efficient allocation with costly value estimation, 2026. URL https://arxiv.org/abs/2608.20316.

Neha Gupta, Harikrishna Narasimhan, Wittawat Jitkrittum, Ankit Singh Rawat, Aditya Krishna Menon, and Sanjiv Kumar. Language Model Cascades: Token-level uncertainty and beyond, 2024. URL https://arxiv.org/abs/2404.10136.

Qitian Jason Hu, Jacob Bieker, Xiuyu Li, Nan Jiang, Benjamin Keigwin, Gaurav Ranganath, Kurt Keutzer, and Shriyash Kaustubh Upadhyay. RouterBench: A benchmark for multi-LLM routing system, 2024. URL https://arxiv.org/abs/2403.12031.

Wenqi Huang, Charley Lee, Leonard Tng, and Serena Ge. DeepSWE: Measuring Frontier Coding Agents on Original, Long-Horizon Engineering Tasks, 2026. URL https://arxiv.org/abs/2607.07946.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code, 2024. URL https://arxiv.org/abs/2403.07974.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can Language Models Resolve Real-World GitHub Issues?, 2023. URL https://arxiv.org/abs/2310.06770.

Wittawat Jitkrittum, Harikrishna Narasimhan, Ankit Singh Rawat, Jeevesh Juneja, Congchao Wang, Zifeng Wang, Alec Go, Chen-Yu Lee, Pradeep Shenoy, Rina Panigrahy, Aditya Krishna Menon, and Sanjiv Kumar. Universal model routing for efficient LLM inference. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 10169–10218, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/113ae3a9762ca2168f860a8501d6ae25-Paper-Conference.pdf.

Rishyanth Kondra, Sanket Mhatre, Akshit Kumar, Evan Chu, Bilal Bakht Ahmad, Ayush Nangia, Rajan Agarwal, Arpan Dasgupta, Animesh Sinha, Bhuvanesh Sridharan, Krupa Dave, Brendan Graham, Guanyu Song, Anirudh Rahul, Wei Hern Lim, Abishek Thangamuthu, Ramneet Singh, Danna Liu, Navid Pour, Calvin Chen, and Justus Mattern. FrontierSWE v2. *Proximal Blog*, 2026. URL https://www.frontierswe.com/blog/v2.

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. LLMRouterBench: A Massive Benchmark and Unified Framework for LLM Routing, 2026. URL https://arxiv.org/abs/2601.07206.

Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From Crowdsourced Data to High-Quality Benchmarks: Arena-Hard and BenchBuilder Pipeline, 2024. URL https://arxiv.org/abs/2406.11939.

Yang Li. Rethinking Predictive Modeling for LLM Routing: When Simple kNN Beats Complex Learned Routers, 2025. URL https://arxiv.org/abs/2505.12601.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. Are Your LLMs Capable of Stable Reasoning?, 2024a. URL https://arxiv.org/abs/2412.13147.

Yueyue Liu, Hongyu Zhang, Yuantian Miao, Van-Hoang Le, and Zhiqiang Li. OptLLM: Optimal Assignment of Queries to Large Language Models, 2024b. URL https://arxiv.org/abs/2405.15130.

Mathematical Association of America. American Invitational Mathematics Examination (AIME), n.d. URL https://maa.org/maa-invitational-competitions/. Accessed September 26, 2026.

Mike A. Merrill, Alexander G. Shaw, Nicholas Carlini, et al. Terminal-Bench: Benchmarking Agents on Hard, Realistic Tasks in Command Line Interfaces, 2026. URL https://arxiv.org/abs/2601.11868.

Stefan Nielsen, Edoardo Cetin, Peter Schwendeman, Qi Sun, Jinglue Xu, and Yujin Tang. Learning to orchestrate agents in natural language with the conductor. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=U23A2BUKYt.

Isaac Ong, Amjad Almahairi, Vincent Wu, Wei-Lin Chiang, Tianhao Wu, Joseph E Gonzalez, M Kadous, and Ion Stoica. RouteLLM: Learning to route LLMs from preference data. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 34433–34448, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/5503a7c69d48a2f86fc00b3dc09de686-Paper-Conference.pdf.

Pranoy Panda, Raghav Magazine, Chaitanya Devaguptapu, Sho Takemori, and Vishal Sharma. Adaptive LLM Routing under Budget Constraints, 2025. URL https://arxiv.org/abs/2508.21141.

Shivam Patel, Neharika Jali, Ankur Mallick, and Gauri Joshi. ProxRouter: Proximity-Weighted LLM Query Routing for Improved Robustness to Outliers, 2025. URL https://arxiv.org/abs/2510.09852.

Long Phan, Alice Gatti, Ziwen Han, et al. Humanity's Last Exam, 2025. URL https://arxiv.org/abs/2501.14249.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A Graduate-Level Google-Proof Q&A Benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Soham Shah and Kumar Shridhar. Select-then-Route: Taxonomy guided routing for LLMs. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 425–441. Association for Computational Linguistics, 2025. doi: 10.18653/v1/2025.emnlp-industry.28. URL https://aclanthology.org/2025.emnlp-industry.28/.

Chenyang Shao, Xinyang Liu, Yutang Lin, Fengli Xu, and Yong Li. Route-and-Reason: Scaling Large Language Model Reasoning with Reinforced Model Router, 2025. URL https://arxiv.org/abs/2506.05901.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. Large Language Model Routing with Benchmark Datasets, 2023. URL https://arxiv.org/abs/2309.15789.

Seamus Somerstep, Felipe Maia Polo, Allysson Flavio Melo de Oliveira, Prattyush Mangal, Mírian Silva, Onkar Bhardwaj, Mikhail Yurochkin, and Subha Maity. CARROT: A Cost Aware Rate Optimal Router, 2025. URL https://arxiv.org/abs/2502.03261.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhu Chen. MMLU-Pro: A More Robust and Challenging Multi-Task Language Understanding Benchmark, 2024. URL https://arxiv.org/abs/2406.01574.

Jason Wei, Nguyen Karina, Hyung Won Chung, Yunxin Joy Jiao, Spencer Papay, Amelia Glaese, John Schulman, and William Fedus. Measuring Short-Form Factuality in Large Language Models, 2024. URL https://arxiv.org/abs/2411.04368.

Fangzhou Wu and Sandeep Silwal. Efficient Training-Free Online Routing for High-Volume Multi-LLM Serving, 2025. URL https://arxiv.org/abs/2509.02718.

Jiaqi Xue, Qian Lou, Jiarong Xing, and Heng Huang. R2-router: A new paradigm for llm routing with reasoning, 2026. URL https://arxiv.org/abs/2602.02823.

Yiqun Zhang, Hao Li, Jianhao Chen, Hangfan Zhang, Peng Ye, Lei Bai, and Shuyue Hu. Beyond GPT-5: Making LLMs cheaper and better via performance-efficiency optimized routing. In *Proceedings of the 2025 7th International Conference on Distributed Artificial Intelligence*, DAI '25, pp. 122–129. Association for Computing Machinery, 2025a. doi: 10.1145/3772429.3772445. URL https://doi.org/10.1145/3772429.3772445.

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. The Avengers: A Simple Recipe for Uniting Smaller Language Models to Challenge Proprietary Giants, 2025b. URL https://arxiv.org/abs/2505.19797.

Zesen Zhao, Shuowei Jin, and Z. Morley Mao. Eagle: Efficient Training-Free Router for Multi-LLM Inference, 2024. URL https://arxiv.org/abs/2409.15518.

Richard Zhuang, Tianhao Wu, Zhaojin Wen, Andrew Li, Jiantao Jiao, and Kannan Ramchandran. EmbedLLM: Learning Compact Representations of Large Language Models, 2024. URL https://arxiv.org/abs/2410.02223.

A ROUTING ACCURACY CEILING

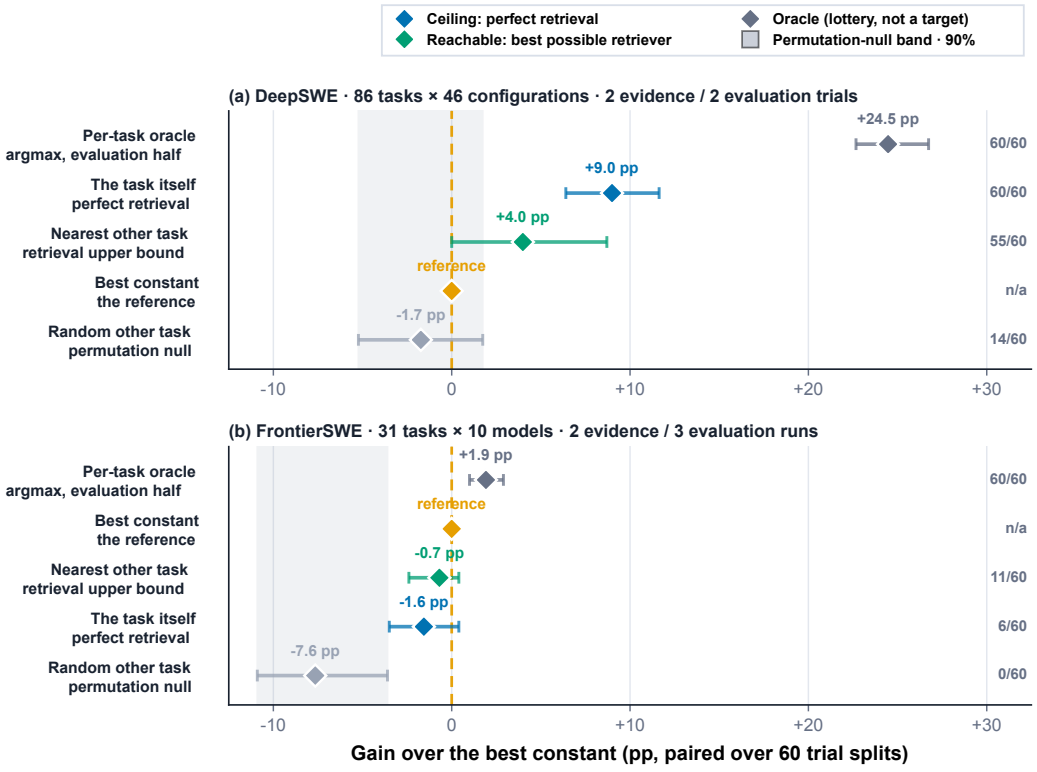


Figure 4: Accuracy gains from different sources of routing evidence.