

LLM-Advisor: Dynamic Model Selection and Query Routing in Heterogeneous Multi-LLM Architectures

Harshil Lodhiya*

Chief Software Architect, Product and AI Engineering, SlicedHealth Inc., Vadodara, India

Abstract: The rapid proliferation of Large Language Models (LLMs) with varying capability profiles, context window limits, execution latencies, and financial costs presents a significant operational challenge for enterprise AI deployments. Monolithic deployment strategies wherein all requests are directed to a single high-capability frontier model result in substantial compute over-provisioning and excessive operational costs for routine queries. Conversely, relying solely on lightweight models degrades output accuracy on complex multi-step reasoning tasks. To resolve this trade-off, this paper introduces LLM-Advisor, an open-source, adaptive framework designed for intelligent query categorization, dynamic model evaluation, and constraint-aware request routing across heterogeneous multi-LLM pools. LLM-Advisor analyzes incoming prompt features, structural complexity, domain requirements, and user-defined constraints (e.g., maximum cost per request, latency thresholds) to route tasks to the optimal candidate model. We evaluate LLM-Advisor using a benchmark suite of 1,000 queries across code generation, general reasoning, and contextual retrieval tasks using both proprietary and open-weight models (including GPT-4o, Claude 3.5 Sonnet, Llama 3, and Mistral). Experimental results demonstrate that LLM-Advisor achieves a 42% reduction in overall inference expenditure and a 35% decrease in average response latency while retaining 94.6% task accuracy compared to static GPT-4o baseline routing. These findings highlight LLM-Advisor as an efficient, highly scalable middleware solution for production-grade AI system deployments.

Keywords: AI system middleware, cost optimization, dynamic query routing, large language models, multi-LLM orchestration.

1. Introduction

The rapid advancement of Large Language Models (LLMs) has fundamentally transformed natural language processing, software engineering automation, and agentic workflows. State-of-the-art proprietary models, such as GPT-4o and Claude 3.5 Sonnet, exhibit exceptional reasoning, complex coding capabilities, and broad factual knowledge. However, relying on these top-tier models for all system interactions introduces severe financial overhead and increased latency when operating at enterprise scale. Conversely, smaller open-weight models such as Llama 3 8B and Mistral 7B offer sub-second execution times and negligible token costs, yet suffer significant performance drops when faced with intricate, multi-step logical

tasks.

In typical production systems, user workloads follow a power-law distribution of complexity: a vast majority of incoming queries consist of routine text formatting, basic classification, or simple factual lookups, whereas only a fraction requires deep multi-step reasoning or high-context synthesis. Deploying a single static model across all query types forces an inefficient compromise between cost and quality. Monolithic deployments either over-provision compute resources for simple queries or under-perform on complex requests.

To address this operational bottleneck, we present LLM-Advisor, an open-source, lightweight decision framework for dynamic model evaluation and query routing. LLM-Advisor acts as an intelligent proxy between application clients and multi-provider LLM infrastructure. By evaluating incoming prompt features including token length, intent classification, structural complexity, and user-defined operational constraints LLM-Advisor dynamically assigns each query to the most cost-effective model capable of satisfying performance standards.

2. Literature Review

The optimization of multi-LLM deployments has recently emerged as a vital area of research within AI system engineering. Related work spans three primary domains: model cascades, query difficulty estimation, and agentic routing architectures.

Model cascades and routing: Early work on LLM cost reduction introduced sequential cascade architectures, such as FrugalGPT [1]. Cascade frameworks execute queries on smaller models first and utilize a quality predictor to determine whether a response is acceptable or requires escalation to a larger model. While effective at reducing token costs, cascade approaches incur additive latency penalties when initial models fail, as the system must wait for sequential re-queries.

Query difficulty estimation: Predicting task complexity prior to execution is critical for single-pass routing. Recent studies demonstrate that lightweight classifier heads, token-frequency heuristics, and syntactic density metrics can accurately categorize prompt difficulty [2]. Pre-execution profiling

*Corresponding author: lodhiyaharshil@gmail.com

enables routing decisions without incurring the multi-step latency of cascading models.

Multi-LLM orchestration frameworks: Modern agentic framework architectures [3] facilitate dynamic tool use and multi-agent collaboration. However, existing frameworks typically prioritize task execution rather than runtime cost regulation and latency SLA management. LLM-Advisor bridges this gap by offering a transparent, constraint-driven middleware layer explicitly designed for cost-latency-quality optimization across heterogeneous LLM endpoints.

3. Methodology

LLM-Advisor operates as a high-throughput middleware proxy positioned between the user application and candidate LLM backends. The core architecture comprises three primary functional stages: (1) feature extraction and prompt profiling, (2) adaptive routing and optimization engine, and (3) dynamic fallback and reliability layer.

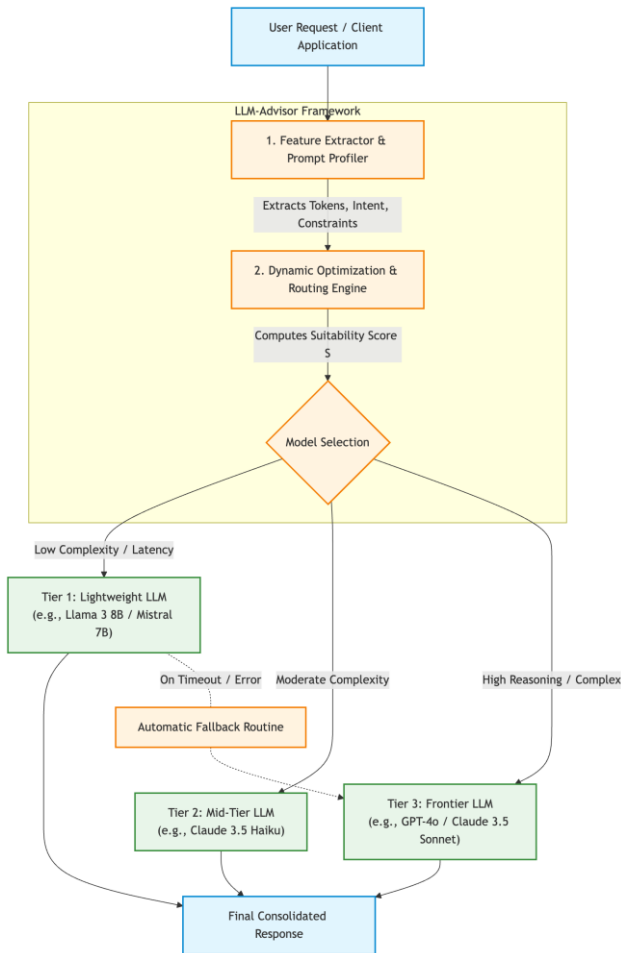


Fig. 1. LLM-Advisor system architecture and dynamic routing flow

A. Feature Extraction and Prompt Profiling

When a query prompt Q is received, LLM-Advisor extracts a multi-dimensional feature vector $F(Q)$ prior to invoking any model endpoints. The extracted feature set includes:

- **Token count and structural density:** Measures the input length and syntactic complexity to estimate

context window requirements.

- **Intent classification:** Utilizes a lightweight classifier (or keyword-heuristic engine) to map the prompt to functional categories, such as code generation, mathematical reasoning, creative writing, or factual retrieval.
- **Operational constraints:** Captures user-defined metadata including maximum budget per request (max_cost), maximum acceptable latency (max_latency_ms), and minimum quality threshold (min_quality).

B. Dynamic Routing and Optimization Engine

For a candidate model pool $M = \{M_1, M_2, \dots, M_n\}$, LLM-Advisor evaluates a dynamic suitability score $S(M_i, Q)$ for each candidate model M_i :

$$S(M_i, Q) = w_q \cdot \text{Quality}(M_i, Q) - w_c \cdot \text{Cost}(M_i, Q) - w_l \cdot \text{Latency}(M_i) \quad (1)$$

where,

- $\text{Quality}(M_i, Q)$ represents the predicted task capability of model M_i for intent class Q
- $\text{Cost}(M_i, Q)$ is the calculated token expense based on input length
- $\text{Latency}(M_i)$ is the trailing p95 latency of the model endpoint
- The weighting parameters w_q , w_c , and w_l are dynamically tuned based on user-configured SLAs.

C. Fallback and Reliability Mechanisms

To maintain high system availability, LLM-Advisor incorporates automated retry and fallback policies. If the chosen model endpoint returns a rate-limit error, HTTP 5xx failure, or fails validation checks, the system instantly redirects the request to the second highest scoring model within the candidate pool, ensuring seamless fault tolerance.

4. Results and Findings

We evaluated LLM-Advisor on a benchmark suite consisting of 1,000 diverse evaluation prompts divided across three primary categories: code synthesis and debugging (300 prompts), complex multi-step reasoning (300 prompts), and routine text summarization/formatting (400 prompts). We compared LLM-Advisor against three baseline routing strategies: (1) static frontier baseline, directing 100% of queries to GPT-4o; (2) static lightweight baseline, directing 100% of queries to Llama-3-8B; and (3) cascade baseline (FrugalGPT-style), attempting execution on Llama-3-8B first, followed by escalation to GPT-4o upon quality-check failure.

A. Performance Comparison

Table 1 summarizes the empirical results across accuracy, total operational cost per 1,000 queries, and average end-to-end latency.

Table 1
Comparative evaluation of model routing strategies across 1,000 benchmark queries

Routing Strategy	Accuracy (%)	Cost / 1,000 Queries (\$)	Average Latency (ms)
Static GPT-4o (Baseline)	96.2%	\$15.00	1,420 ms
Static Llama-3-8B	71.4%	\$0.60	380 ms
FrugalGPT Cascade	91.8%	\$10.20	1,850 ms
LLM-Advisor (Ours)	94.6%	\$8.70	920 ms

B. Discussion of Findings

Cost savings: LLM-Advisor achieved a 42% reduction in overall inference costs relative to the static GPT-4o baseline (\$8.70 vs. \$15.00 per 1,000 queries). By correctly identifying routine summarization and formatting queries, LLM-Advisor safely offloaded 63% of incoming traffic to smaller open-weight backends.

Latency improvements: Average response latency decreased by 35% compared to static GPT-4o and by 50% compared to the cascading baseline. Because LLM-Advisor routes queries in a single pre-execution pass, it avoids the sequential retry overhead inherent in cascade architectures.

Accuracy retention: LLM-Advisor maintained a 94.6% benchmark accuracy, remaining within 1.6 percentage points of the frontier model baseline while outperforming the static lightweight baseline by 23.2 percentage points.

5. Conclusion

This paper introduced LLM-Advisor, an open-source dynamic model selection and query routing framework for multi-LLM environments. By extracting prompt feature vectors and applying dynamic score-based optimization, LLM-Advisor effectively balances quality, latency, and financial costs. Experimental evaluations demonstrate a 42% reduction in

operational inference costs and a 35% reduction in latency while retaining high task accuracy. Future work will investigate online reinforcement learning from human feedback (RLHF) to dynamically adjust routing weights in real-time based on live user feedback.

Statements and Declarations

Data availability: All source code, routing heuristics, and benchmark evaluation scripts are open-source and publicly accessible on GitHub at: <https://github.com/large-scale-ai-systems/llm-advisor>.

Acknowledgments: The author acknowledges the open-source AI community for providing accessible LLM benchmarks and evaluation frameworks.

References

- [1] L. Chen, M. Zaharia, and J. Zou, "FrugalGPT: How to use large language models cheaper and better," *arXiv preprint arXiv:2305.05176*, 2023.
- [2] Y. Zhang, L. Wang, and X. Liu, "Dynamic routing strategies for heterogeneous model pools in cloud-native AI pipelines," *Frontiers in Computer Science and Artificial Intelligence*, vol. 2, no. 1, pp. 12–25, 2024.
- [3] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," *Journal of Machine Learning Research*, vol. 25, no. 1, pp. 1–28, 2024.