# Multi-LLM Routing on Use Case and User Preferences

**Github Link**

https://github.com/Kumaresh4321/CS_769_Project.git

# Table of Contents

## 1. Project Title, Members, and Contributions

Our project is titled "Multi-LLM Routing on Use Case and User Preferences". This project will be an extension to the paper "RouterBench: A Benchmark for Multi-LLM Routing System". Our members are Abhijnya Menakur, Kumaresh Suresh Babu, and Kincannon Wilson.

In addition, five members from the CS 839 course on Foundation Models are also contributing to the project. These members are Yancheng Zhu, Jiawei Zhang, Xinyue Lin, Tianyi Li, Fahad Touseef, and Aydan Bailey.

For this report, Abhijnya conducted the literature survey, Kumaresh reimplemented the RouterBench results, and Kincannon completed the other sections. The five members from 839 did not contribute to the report.

## 2. Project Overview

As companies like OpenAI, Google, and Anthropic continue to expand their API offerings, users are faced with a challenging landscape of model options, each with varying performance, cost, and latency. Selecting the best model for a specific use case requires not only understanding performance benchmarks but also navigating frequent updates, emerging models, and complex APIs.

Publicly available benchmarks, such as GSM8K[1] for math problems and MMLU[2] for reasoning tasks, provide useful performance insights but are limited. Not all models report on each benchmark, and users often must evaluate technical papers and experiment with multiple APIs to find the best model for their needs, which can be time-consuming and costly.

Multi-LLM routing systems have begun to simplify model selection by choosing the best model based on prompt, price, and latency. Recent work like RouterBench[3] has established a benchmark for these systems, evaluating 400,000 inference outcomes to assess routing efficiency across models. Martian, the organization behind RouterBench, provides a manual experimentation platform for users, yet it still demands substantial technical knowledge and effort.

Our project will build on the principles of RouterBench to create an advanced routing system that automatically selects the best model for each user based on specific use cases and preferences without manual testing or prior knowledge about available models and model providers. By improving accessibility, efficiency, and adaptability, our approach will make sophisticated LLM capabilities available to a wider audience, driving innovation and expanding LLM application potential across industries.

## 3. Literature Survey

The availability of diverse Large Language Models (LLMs) from providers like OpenAI, Google, Anthropic has led to a demand for efficient milti-LLM routing systems that can optimize model selection based on user needs for cost, latency, and performance. Our project, “Multi-LLM Routing on Use Case and User Preferences”, builds upon foundational work, particularly RouterBench, and incorporates innovative ranking and cost efficient strategies inspired by recent literature.

1. **RouterBench: A Benchmark for Multi-LLM Routing Systems**
   *RouterBench* (Hu et al., 2024) offers a standardized framework for evaluating multi-LLM routing, measuring cost, latency, and task-specific performance across diverse LLMs. With over 405,000 inference samples, RouterBench enables detailed comparison of routing strategies under different conditions and provides a platform for understanding the trade-offs between cost and performance. Our project extends these principles to automatically route models based on user-defined preferences, making the model selection process adaptable to real world applications.

2. **LLM-as-a-Judges with MT-Bench and CHatbot Arena**
   Zheng et al. (2023) introduce “LLM-as-a-Judge,” an innovative approach where LLMs themselves serve as judges to score model outputs, simulating human evaluation. This judge model enables automated, scalable evaluations that closely align with human preferences, achieving over 80% agreement with human judgment. Our model incorporates the similar judge model to rank candidate LLms for specific user requirements, enabling iterative improvement in routing decisions by continuously comparing the system’s automated rankings with human expectations.

3. **FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance**
   *FrugalGPT* (Chen et al., 2023) proposes a sequential routing strategy that progressively queries models until a quality threshold is met, significantly lowering inference costs. This cost-sensitive approach aligns with our goal of optimizing model selection based on budget and performance requirements, allowing our routing system to dynamically prioritize models that meet user-defined criteria.

4. **Blending Is All You Need: Cost-Effective Alternatives to Large LLMs**
   Lu et al. (2024) explore "blending" smaller models to rival the performance of larger ones like ChatGPT. By combining models with 6B and 13B parameters, they achieve high efficacy at reduced computational costs. This blending strategy is relevant to our project’s

aim to utilize a range of model sizes and costs, providing a flexible, efficient multi-LLM routing system that doesn't rely solely on large, resource-intensive models.

Together, these works provide a foundation for our project. By integrating RounterBench's benchmarking, LLM-as-a-Judge's automated ranking, FrugalGPT's cost sensitive methods, and the efficiency of blending the task specific routing, our system will offer a flexible, adaptive multi-LLM routing solution. This approach makes LLM capabilities  more accessible and optimized for various user defined preferences across applications in text, audio, image and video processing.

## 4. Reimplementation of Related Work

We implemented the RouterBench setup, testing the routing strategies: Zero, KNN, MLP and cascading routers across various datasets. Our evaluation results, summarized in the cost-performance graphs, illustrate the efficiency and performance balance each router achieves under different conditions.

- MTBench: In the MTBench dataset, the Zero router showed a balanced cost-performance ratio, closely aligning with the KNN and MLP routers. This dataset demonstrates the effectiveness of simpler routers in delivering reliable performance at a lower cost threshold.
- Grade School Math: For the Grade School Math dataset, the MLP and zero routers performed comparably, with KNN showing slightly higher costs but similar accuracy. This dataset, which requires logical reasoning and arithmetic, highlighted the adaptability of MLP and Zero routers in managing Performance without excessive cost increments.
- MMLU: The MMLU dataset, which assesses general world knowledge, showed all routers performing well with minimal cost, with the KNN may be better suited for tasks requiring a diverse knowledge base.
- ARC-Challenge: In the ARC-Challenge, the Zero and MLP routers achieved high performance with minimal cost. This shows Zero and MLP routers are better for domain specific reasoning tasks, balancing cost and accuracy effectively.
- Winogrande:On the Winogrande dataset, which tests commonsense reasoning, the KNN router exhibited a slight advantage, achieving higher performance relative to cost compared to the other routers. This suggests that KNN routing is more suitable for tasks with nuanced reasoning requirements.
- HellaSwag: The HellaSwag dataset revealed that all routers reached similar performance levels with comparable costs, indicating that tasks involving commonsense understanding are generally robust across different routing strategies.

- MBPP:For the MBPP dataset, the zero router achieved best cost-performance trade-off, followed closely by the KNN and MLP routers. This suggests that the zero router is particularly cost-effective for technical and code-based tasks.

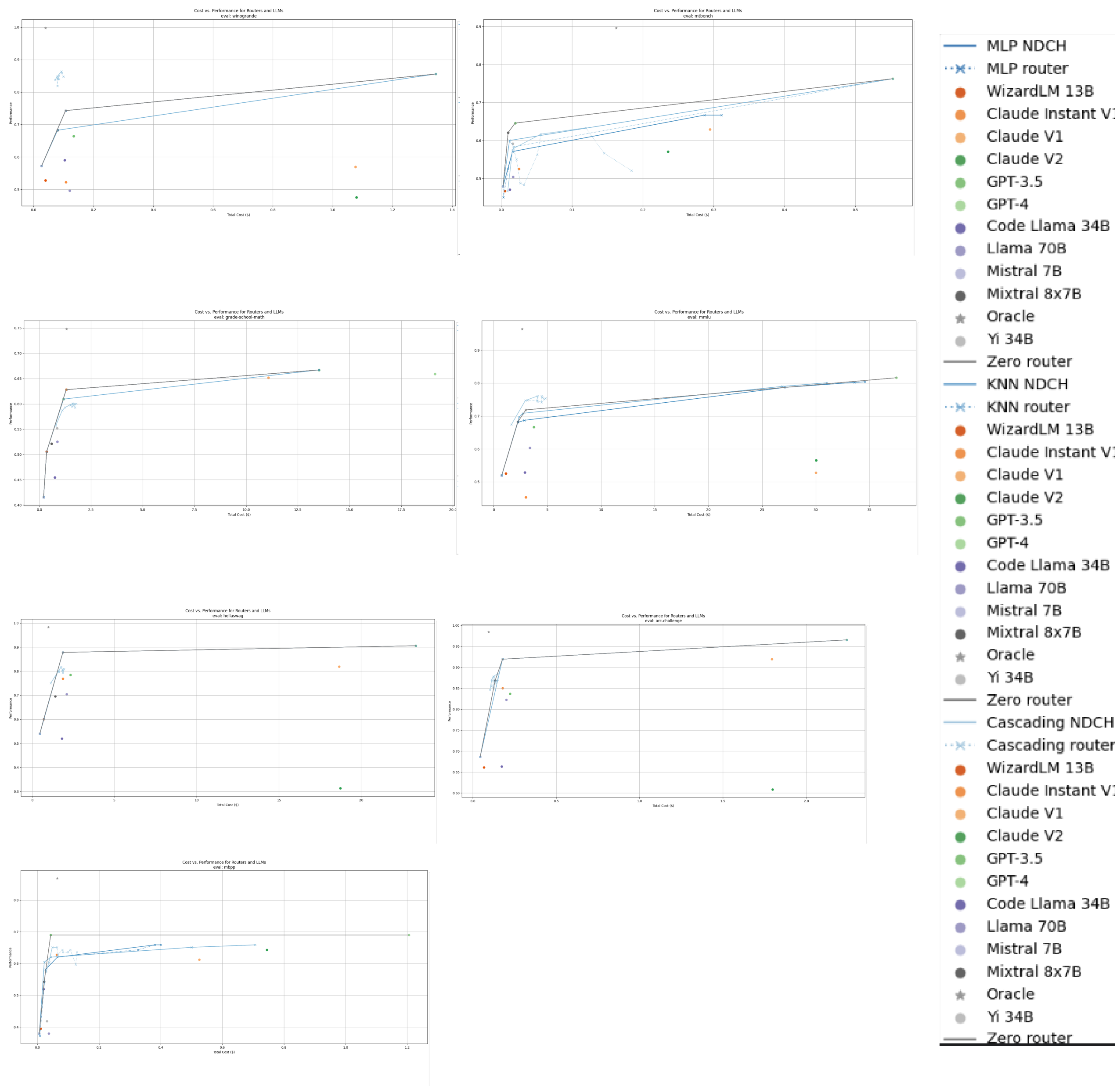

*Figure 1:* Each of these evaluations is presented in the accompanying cost vs performance graphs, which illustrate the non-decreasing convex hull (NDCH) for each router configuration (KNN, MLP, ZERO and Cascading routers). The NDCH lines highlight the optimal cost-performance points, providing visual insights into the most effective configuration across the routers and datasets.

Our findings underscore that different routing strategies are suited to different types of tasks:

- Zero Router: Offers a cost-effective option for simpler tasks that do not require complex routing decisions.
- KNN and MLP Routers: Provide steady and reliable performance, making them suitable for general applications where consistent outcomes are prioritized over cost.
- Cascading Router: Shows particular strength in handling tasks of varied complexity. By adopting a tiered approach, it is able to escalate to more advanced models only when necessary, resulting in efficient use of resources and cost savings.

The inclusion of the cascading router illustrates a dynamic model selection strategy that adapts to task complexity, helping to minimize costs while maintaining high accuracy. This adaptive approach is especially advantageous in real world applications with diverse input types, where some queries can be addressed by simpler models while others demand more sophisticated capabilities.

Our reimplementation confirms that RouterBench's findings hold across different datasets, no single routing strategy is universally optimal. Instead, the choice of routing strategy depends on the specific requirements of the task and cost considerations. While Zero router provides a simple and economical solution, the Cascading router's adaptability to task complexity makes it valuable in scenarios requiring flexibility and efficiency.

## 5. Methods Beyond Existing Work

The existing Multi-LLM routing methods largely rely on static criteria, focusing primarily on cost while requiring predefined model selection for routing. These approaches offer limited flexibility as users must decide which models to include in their routing pool manually, restricting the dynamic selection of models and missing key performance metrics such as latency. Our approach aims to enhance these methodologies by introducing a framework that incorporates latency as an essential selection criterion. This will allow for an adaptive routing system that autonomously optimizes for latency, cost, and performance preferences, tailored to the specific requirements outlined by users. By automating the model selection process, our method eliminates the need for manual configuration, enhancing user efficiency and optimizing model usage based on real-time performance data.

In addition to improving LLM routing, our approach seeks to extend these principles across multiple modalities beyond language models. We aim to incorporate a wider array of models, including those for video analysis, image processing, and multimodal inputs, enabling the routing system to serve applications that require more than text-based analysis. By integrating models from diverse modalities, our solution will be able to handle more complex, cross-modal tasks seamlessly, making it adaptable for various industry applications. This multimodal expansion will not only widen the practical use cases but also provide an integrated platform for selecting the optimal model across different data types and sources, a feature currently absent in existing systems.

Furthermore, our method will automate the system's updates and benchmarking processes to accommodate the rapidly evolving LLM landscape. Given the frequency with which new models are introduced or updated, our approach will include mechanisms for periodic reassessment of model costs, latency, and task-specific performance metrics. This automated reassessment will ensure that the system dynamically adjusts its model selection pool to maintain optimal routing without requiring manual intervention.

## 6. Dataset

Our work would first involve the creation of a dataset of 35+ tasks. Examples include translation, data cleaning/filtering, text summarization, and other real-world examples. We plan to have example tasks across a range of modalities, including text, image, audio, and video. Each modality, along with the categories of task we plan to consider in that modality are listed below:

**TEXT**
- Language translation
- Summarization
- Code analysis
- Sentiment analysis
- Question answering
- Document/template creation
- Scheduling/planning
- Text generation (general)

**AUDIO**
- Transcription
- Speech from text

**IMAGE**
- Image editing
- Image generation from text
- Image captioning
- Recognize text within image

**VIDEO**

- Generation
- Summarization
- Captioning

Each task will include a unique identifier, a description, a modality, a category, one or more examples of prompt/desired output pairs for the task, a natural language description of the user's preferences for latency/cost/performance, and numeric scores for a given user's price/latency/performance preferences for that use case. We will generate this dataset by hand, taking care to create each task according to a realistic user persona and application.

## 7. Experiments

We plan to run experiments to answer the following questions:

- How well does our automated system agree with human judgment?
- Does our automated system agree with human judgment best when doing single, pairwise, or leaderboard/win-rate comparisons?
- Does specifying preferences for price/latency/performance in a numeric form outperform natural language preferences?
- How does prompt structure affect the judge model?
- What is the best method for extracting relevant information from the judge model's output?
- For a given use case, we could have an LLM determine which benchmarks are relevant. Then, would including the performance of a candidate model on relevant benchmarks improve the evaluations made by the judge model?
- Does the performance of the automated system vary by modality/category compared to human feedback?

Although we have not determined exact details, we hope to run experiments using at least 10 models from popular providers including OpenAI, Anthropic, Meta, Mistral, and Salesforce. We will include models representing a wide range of quality, cost, and latency in order to test the performance of our routing system. We plan to generate model output using OpenRouter's API, which provides access to multiple models using one API key.

## 8. Proposed Ideas and Project Plan

We plan to complete the following tasks in the order described:

- Create the dataset of tasks
- Create the list of models including price, latency, and wrapper functions for easily getting their output from an API
- Create the human rankings for all models for each task
- Create the prompts for how the judge model will be asked to evaluate models
- Create a system for ranking models on a specific task based on price/latency/performance. We will create systems based on the win-rate compared to a judge model, pairwise comparison, and single comparison.
- Decide on metric(s) for comparing ranking similarity (potential options include Intersection Count, Jaccard Similarity, and others)
- Run experiments listed in the section above
- Time permitting, we will begin to work on methods for efficient updates following changes to user prompts, user preferences, or model cost/latency/performance

The work will be evenly divided among the group members. Please note that this report details all of the potential ideas we have for our presentation and project and that we may not be able to work on all of them within the time constraints of the course.

## 9. Citations

1. Cobb, A., et al. (2022). "GSM8K: A Grade School Math Dataset." Proceedings of the NeurIPS Workshop on Foundation Models.
2. Hendrycks, D., et al. (2021). "Massive Multitask Language Understanding (MMLU)." arXiv preprint.
3. Zhang, Y., et al. (2023). "RouterBench: Evaluating Multi-LLM Routing Systems." Proceedings of the International Conference on Machine Learning (ICML).
4. Hu, Q. J., Bieker, J., Li, X., et al. (2024). "RouterBench: A Benchmark for Multi-LLM Routing Systems." arXiv preprint arXiv:2403.12031.
5. Zheng, L., Xu, H., Zhu, Y., et al. (2023). "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." arXiv preprint arXiv:2306.05685.
6. Chen, L., Zaharia, M., & Zou, J. (2023). "FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance." arXiv preprint arXiv:2305.05176.
7. Lu, X., Liu, Z., Liusie, A., et al. (2024). "Blending Is All You Need: Cheaper, Better Alternative to Trillion-Parameter LLM." arXiv preprint arXiv:2401.02994.