Efficient Similarity Index Based Task Routing for LLM

By

Liew Zheng Xian

A REPORT

SUBMITTED

TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology
(Kampar Campus)

JUNE 2025

ACKNOWLEDGEMENTS

I would like to express my appreciation to my supervisor, Dr Aun Yichiet who has given me this bright opportunity to engage in a generative AI project.

Finally, I want to give my special thanks to my parents and my family for their love, support, and continuous encouragement throughout the course.

COPYRIGHT STATEMENT

ABSTRACT

Large Language Models (LLMs) represent a significant innovation within the field of Generative AI (GenAI), yet their practical deployment is hampered by a critical trade-off between the high performance of large models and the cost-efficiency of smaller ones. This research addresses this challenge by designing and evaluating an intelligent LLM routing system to optimize both cost and performance. Four distinct routing methods, including standalone classifiers like XGBoost and a hybrid KNN+XGBoost model, were evaluated on their ability to dynamically select the most suitable LLM for queries across diverse benchmarks: conversational (MT-Bench), mathematical reasoning (GSM8K), and multi-domain (MMLU). The results demonstrate that the proposed routing framework achieves significant cost savings, reducing expenses by up to 2.76× on MT-Bench with minimal impact on response quality. The hybrid KNN+XGBoost model proved most robust, particularly on the challenging MMLU benchmark where other models failed, simpler classifiers struggled in specialized domains. Ultimately, this project validates LLM routing as a powerful strategy for cost optimization.

Area of Study: Generative Artificial Intelligence

Keywords: Large Language Models, Generative Artificial Intelligence, LLM Routing, Machine Learning, Similarity Based Index Routing

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

Gen AI	Generative Artificial Intelligence
LLM	Large Language Model
GAN	Generative Adversarial Network
API	Application Programming Interface
GPT	Generative Pretrained Model
FYP	Final Year Project
JSON	JavaScript Object Notation
CPT	Call Performance Threshold
PGR	Performance Gap Recovered
APGR	Average Performance Gap Recovered
MMLU	Massive Multitask Language Understanding
GSM8K	Grade School Math 8K
kNN	K Nearest Neighbour

CHAPTER 1 - Project Background

1.1 Introduction

In recent years, Generative Artificial Intelligence (Gen AI) has gained significant traction across various domains, including natural language processing, text-to-image generation, and text-to-video content creation. Gen AI models, such as Generative Adversarial Networks (GANs), diffusion models, and Transformer-based models, have demonstrated impressive capabilities in generating high-quality data, from images and text to more complex multi-modal outputs. The creation of these models has opened many possibilities for the future of AI advancement.

At the same time, Large Language Models (LLMs) have rapidly advanced in recent years, delivering state-of-the-art performance across tasks such as conversation, summarization, code generation, and reasoning. While larger models achieve superior accuracy and reasoning ability, their deployment is costly in terms of computation and latency. Conversely, smaller models are more affordable but often underperform on complex tasks. This is because larger LLMs contain billions of parameters, requiring powerful GPUs, large memory, and complex inference pipelines that generate tokens sequentially, leading to slower responses and greater operational expenses. In contrast, smaller LLMs are faster, cheaper to deploy and better suited for high-volume or domain-specific tasks, making them more practical for enterprises and real time applications.

This trade-off between capability and efficiency has motivated LLM routing, the process of directing each query to the most suitable model. By routing simpler tasks to lightweight models and reserving more complex tasks for powerful ones, routing frameworks promise to optimize both cost and performance. A notable recent development is the automatic model selection mechanism in GPT-5, where a built-in router dynamically chooses the most appropriate model variant per query [1].This demonstrates how routing can operate at scale in real-world applications, inspiring research into more generalizable routing solutions.

1.2 Research Objectives

The main objective of this project is to effectively route user task prompts to LLMs to minimize cost. The project shall achieve the following sub-objectives:

- To design and evaluate four routing methods capable of efficiently selecting the most suitable large language model (LLM) for each query across diverse domains.
- To analyse the trade-offs between accuracy, and cost in LLM routing, and to demonstrate that the proposed approach achieves significant cost savings on popular benchmarks with minimal impact on response quality.

1.3 Problem Statement and Motivation

The diversity in LLM capabilities and resource demands poses a deployment dilemma: using a high-capacity model for all queries ensures quality but is inefficient and expensive; using smaller models lowers cost and latency but risks poor responses for complex tasks. Current approaches rarely provide unified, flexible solutions that generalize across domains and model families.

The introduction of automatic model selection in GPT-5 has demonstrated the potential of routing mechanisms to balance computational efficiency and task performance. Nonetheless, this feature is tightly coupled to the OpenAI ecosystem and remains largely opaque in its operation. Moreover, the rollout of GPT-5's automatic routing has been met with notable public criticism. Users expressed dissatisfaction with the removal of manual model selection, citing reduced transparency, diminished control, and workflow disruptions as key concerns[2], [3] .In response to sustained user backlash, OpenAI reintroduced access to legacy models (e.g., GPT-4o) and implemented optional routing modes ("Auto," "Fast," and "Thinking") [4], reflecting the tension between automation and user agency in LLM deployment .These developments highlight the limitations of ecosystem-specific and opaque routing strategies, revealing the importance of designing more transparent, flexible, and generalizable frameworks. Recent studies [5], [6], [7], [8], [9] have begun exploring cross-model routing approaches that operate across heterogeneous LLMs, suggesting a research direction toward scalable, model-agnostic routing systems capable of adapting to evolving model landscapes and diverse application demands. Hence, this research on similarity index-based task routing for LLMs is conducted to explore different routing methods to reduce cost.

1.4 Project Scope and Directions

This project's scope is centered on the design, implementation, and empirical evaluation of a task router for Large Language Models (LLMs). The primary goal is to create a task router that can intelligently and automatically allocate computational resources by selecting from a strong and weak LLM to optimize the trade-off between performance and cost.

Design and Implementation of Routers: To thoroughly investigate the routing problem, the project implements and compares four distinct methods. These include direct supervised classifiers (XGBoost and Logistic Regression) trained on prompt embeddings, a non-parametric similarity model (KNN) that relies on a majority vote from similar prompts, and an advanced hybrid model that combines both approaches by enriching embeddings with similarity-based features before classification.

Evaluation Methodology: The project adheres to a rigorous evaluation framework consistent with the RouteLLM paper so that a comparison can be made with the result of the paper. This involves benchmarking all implemented models on diverse, unseen test datasets (MT-Bench, GSM8K, MMLU) and using a standardized set of efficiency metrics (CPT(50%)/CPT(80%) and APGR).

1.5 Contributions

1. Provide the implementation of four routing methods that select the LLM for each query efficiently.
2. Provide methodology to analyse routing efficiency and conduct experiments across domains to evaluate accuracy, and cost trade off.

3. Demonstrated routing approaches that can achieve cost savings up to 2.76 saving on multiple popular benchmarks with minimal trade-off on the response quality.

The primary contribution of this work is the design of a routing framework that enables efficient selection of LLMs for each query. In addition, a systematic analysis of routing efficiency is conducted through experiments across multiple domains, evaluating performance trade-offs in terms of accuracy and cost. The results demonstrate that the proposed routing approach achieves substantial cost savings on widely used benchmarks, while maintaining competitive response quality. Collectively, these contributions establish a foundation for scalable and reliable deployment of different LLMs in practice.

1.6 Project Organization

The project report includes five chapters. Chapter 1 introduces the problem statement, motivation, research objectives, and contributions this project will bring. Chapter 2 will show literature reviews of relevant generative AI models, related works and benchmarks. Chapter 3 details the proposed solution and approach while describing the high-level architecture overview of the solution. Chapter 4 will describe the experimental work. Chapter 5 will show the evaluation of the results of the experiments. Chapter 6 will describe the conclusion from this project.

CHAPTER 2 - Literature Review

2.1 Existing Generative AI Models

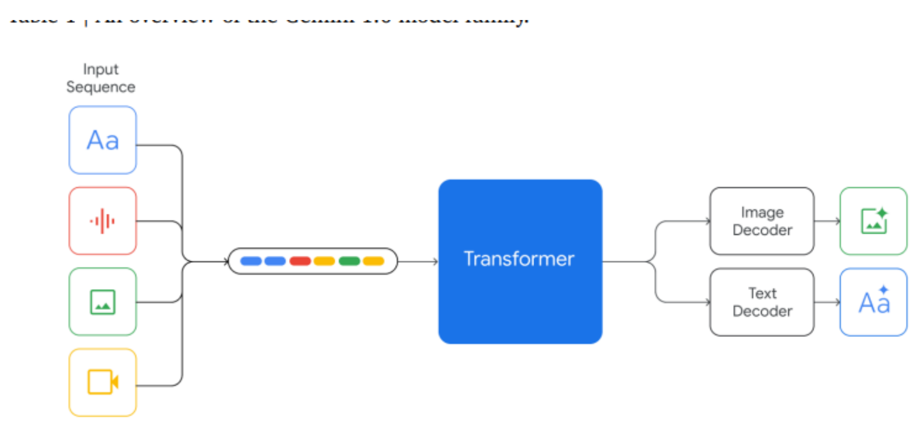
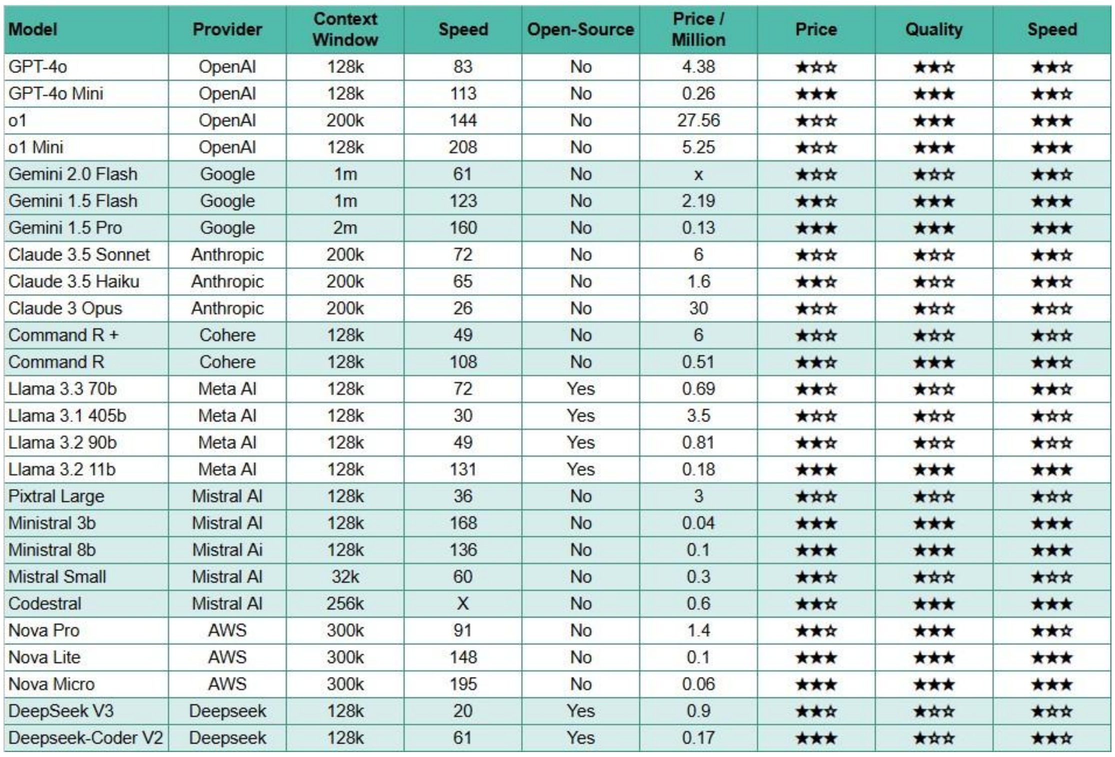


Figure 2.1.1.1 Gemini 1.0 model family

Gemini Flash models leverage the Transformer architecture, often incorporating Mixture-of-Experts for efficient processing [10].These Mixture-of-Experts models divide the network into smaller, specialized expert pathways, selectively activated based on the input to improve efficiency. Gemini Flash is also a multimodal model, accepting diverse inputs like text, code, images, audio, and video. Key features the ability to utilize native tools such as structured output mode which let it always output text in JSON format.



Model	Provider	Context Window	Speed	Open-Source	Price / Million	Price	Quality	Speed
GPT-4o	OpenAI	128k	83	No	4.38	★☆☆	★★☆	★★☆
GPT-4o Mini	OpenAI	128k	113	No	0.26	★★★	★★★	★★☆
o1	OpenAI	200k	144	No	27.56	★☆☆	★★★	★★★
o1 Mini	OpenAI	128k	208	No	5.25	★☆☆	★★★	★★★
Gemini 2.0 Flash	Google	1m	61	No	x	★☆☆	★☆☆	★★☆
Gemini 1.5 Flash	Google	1m	123	No	2.19	★★☆	★★★	★★★
Gemini 1.5 Pro	Google	2m	160	No	0.13	★★★	★★★	★★★
Claude 3.5 Sonnet	Anthropic	200k	72	No	6	★☆☆	★☆☆	★★☆
Claude 3.5 Haiku	Anthropic	200k	65	No	1.6	★★☆	★☆☆	★★☆
Claude 3 Opus	Anthropic	200k	26	No	30	★☆☆	★☆☆	★☆☆
Command R +	Cohere	128k	49	No	6	★☆☆	★☆☆	★☆☆
Command R	Cohere	128k	108	No	0.51	★★☆	★★★	★★☆
Llama 3.3 70b	Meta AI	128k	72	Yes	0.69	★★☆	★☆☆	★★☆
Llama 3.1 405b	Meta AI	128k	30	Yes	3.5	★☆☆	★☆☆	★☆☆
Llama 3.2 90b	Meta AI	128k	49	Yes	0.81	★★☆	★☆☆	★☆☆
Llama 3.2 11b	Meta AI	128k	131	Yes	0.18	★★★	★★★	★★★
Pixtral Large	Mistral AI	128k	36	No	3	★☆☆	★☆☆	★☆☆
Ministral 3b	Mistral AI	128k	168	No	0.04	★★★	★★★	★★★
Ministral 8b	Mistral Ai	128k	136	No	0.1	★★★	★★★	★★★
Mistral Small	Mistral AI	32k	60	No	0.3	★★☆	★☆☆	★☆☆
Codestral	Mistral AI	256k	X	No	0.6	★★☆	★★★	★★★
Nova Pro	AWS	300k	91	No	1.4	★★☆	★★★	★★☆
Nova Lite	AWS	300k	148	No	0.1	★★★	★★★	★★★
Nova Micro	AWS	300k	195	No	0.06	★★★	★★★	★★★
DeepSeek V3	Deepseek	128k	20	Yes	0.9	★★☆	★☆☆	★☆☆
Deepseek-Coder V2	Deepseek	128k	61	Yes	0.17	★★★	★☆☆	★★☆

Figure 2.1.1.2 shows the comparison of LLM on price, quality and speed

In the current landscape, there are many closed source and open source generative AI models available on the internet. Notably, Gemini Flash and Pro model offer largest context window (1 million for Flash models and 2 million for Pro model) at a very low price.

2.2 Related Works

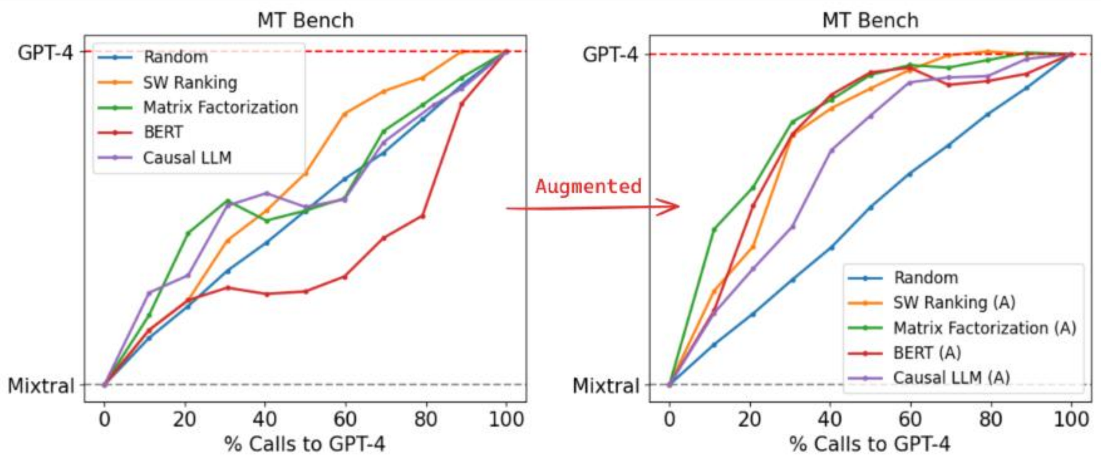


Figure 2.2.1 shows data augmentation improves performance of RouteLLM's router

RouteLLM, represents a significant advancement in the field by introducing a principled framework specifically designed for binary routing between strong and weak LLMs. The methodology centers on training router models using human preference data from Chatbot Arena, combined with sophisticated data augmentation techniques which enhance performance across diverse scenarios as shown in Figure 2.2.1.

RouteLLM evaluates four distinct router architectures: matrix factorisation, similarity-weighted ranking, BERT-based classification, and causal LLM-based routing. Experimental results [11] show cost reductions of over 85% on MT-Bench, 45% on MMLU, and 35% on GSM8K while achieving 95% of GPT-4's performance. Particularly noteworthy is the framework's superior performance compared to commercial routing solutions, achieving comparable results while being over 40% cheaper.

TensorOpera Router (TO-Router), proposed by [7], presents a production-oriented solution to the *multi-LLM routing trilemma* balancing query execution time, monetary cost, and model performance. The framework integrates multiple LLM experts into a unified pipeline through a three-phase methodology like in Figure 2.2.2.

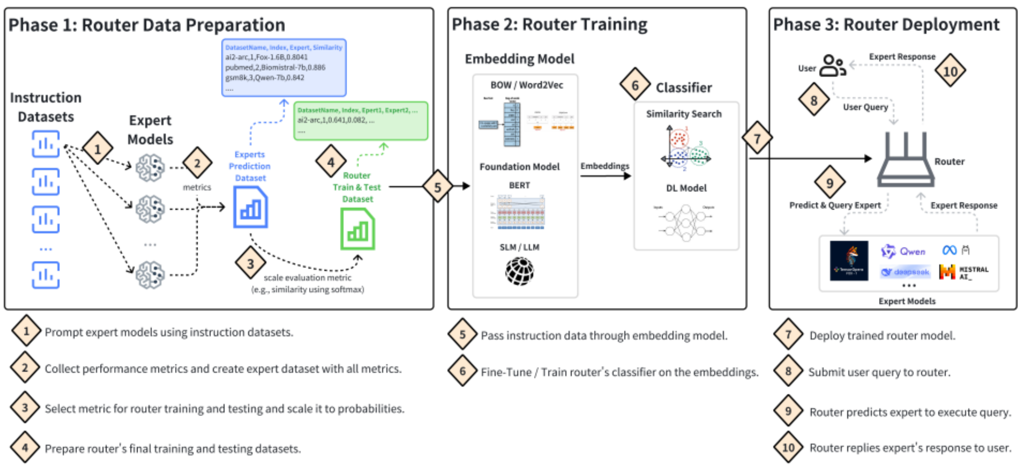

Phase 1: Router Data Preparation
Instruction Datasets
Expert Models
1
2
metrics
DatasetName, Index, Expert, Similarity
ai2-arc,1,Fox-1.6B,0.8041
pubmed,2,Biomistral-7b,0.886
gsm8k,3,Qwen-7b,0.842
....
Experts Prediction Dataset
DatasetName, Index, Epert1, Expert2, ...
ai2-arc,1,0.641,0.082, ...
....
4
Router Train & Test Dataset
3
scale evaluation metric (e.g., similarity using softmax)
5
Phase 2: Router Training
Embedding Model
BOW / Word2Vec
Foundation Model
BERT
SLM / LLM
Embeddings
6
Classifier
Similarity Search
DL Model
7
Phase 3: Router Deployment
User
Expert Response
10
8
User Query
Router
9
Predict & Query Expert
Expert Response
Qwen
deepseek
MISTRAL AI_
Expert Models
1 Prompt expert models using instruction datasets.
2 Collect performance metrics and create expert dataset with all metrics.
3 Select metric for router training and testing and scale it to probabilities.
4 Prepare router's final training and testing datasets.
5 Pass instruction data through embedding model.
6 Fine-Tune / Train router's classifier on the embeddings.
7 Deploy trained router model.
8 Submit user query to router.
9 Router predicts expert to execute query.
10 Router replies expert's response to user.

Figure 2.2.2 Overview of TensorOpera Router's three phase methodology

In the data preparation phase, expert models are used to generate training datasets with key metrics such as negative log-likelihood, BERT similarity scores, inference time, and token counts. A novel contribution is the use of temperature-scaled softmax functions on BERT similarity scores to generate soft labels, enabling the router to learn expert ranking rather than rely on hard classifications.

During router training, four approaches are evaluated: Random-Router, k-Nearest Neighbors Router, MLP-Router, and BERT-Router. Among these, BERT-Router consistently achieves superior results across evaluation dimensions.

In the deployment phase, the router functions as a standalone endpoint for query processing, including tokenization, encoding, classification, and expert selection. Reported improvements include up to 40% higher query efficiency, 30% cost reduction, and 10% performance gains compared to using standalone expert models.

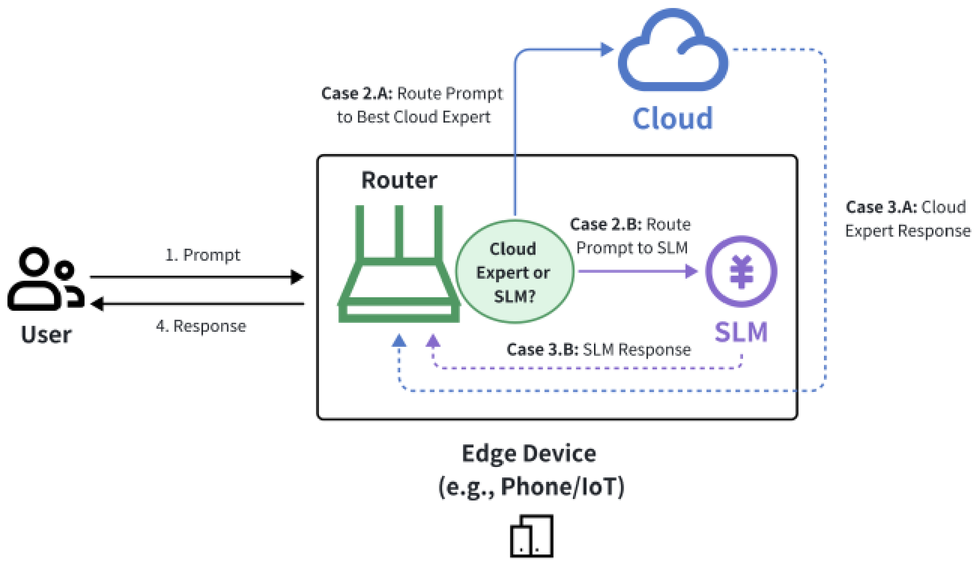

Case 2.A: Route Prompt to Best Cloud Expert
Cloud
Router
Case 2.B: Route Prompt to SLM
Cloud Expert or SLM?
Case 3.A: Cloud Expert Response
1. Prompt
4. Response
User
SLM
Case 3.B: SLM Response
Edge Device (e.g., Phone/IoT)

Figure 2.2.3 TensorOpera Router's proposed edge-to-cloud collaborative routing

Evaluation across four benchmarks: Ai2-ARC, GSM8K, MBPP, and PubMedQA using seven expert models demonstrates that BERT-Router achieves near-optimal routing performance, significantly outperforming naive baselines. Furthermore, TO-Router proposes the idea of edge-to-cloud collaborative routing, dynamically deciding whether queries are processed locally by small models or routed to more powerful cloud-based experts.

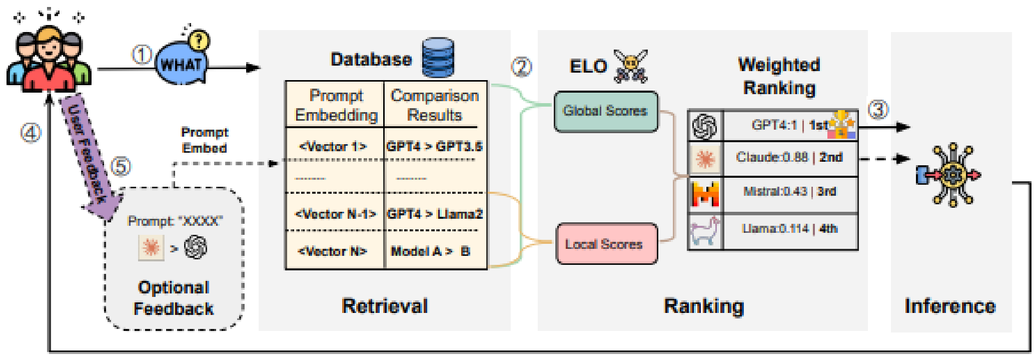


Figure 2.2.4 Eagle's ELO ranking based LLM router

Eagle, proposed by [6], introduces a training-free routing framework designed for scalability and real-time adaptation in high-volume online environments as shown in figure 2.2.4. Unlike training-intensive methods, Eagle enables immediate deployment by leveraging a dual global–local ELO ranking system. The global module assesses general LLM capabilities across diverse tasks, while the local module evaluates specialized expertise for specific query types, allowing for balanced and adaptive routing decisions.

Borrowing from competitive gaming, the ELO ranking mechanism dynamically updates model performance based on user feedback and query outcomes, ensuring continuous adaptation to evolving conditions. Experimental results highlight Eagle's effectiveness, showing up to 23.52% improvements in AUC, 1/20 initialization time, and 100–200× faster incremental updates compared to baseline methods.

By eliminating the need for extensive training data and heavy preparation, Eagle offers a lightweight yet adaptive routing solution well-suited for real-world, large-scale online serving scenarios where efficiency and rapid responsiveness are essential.

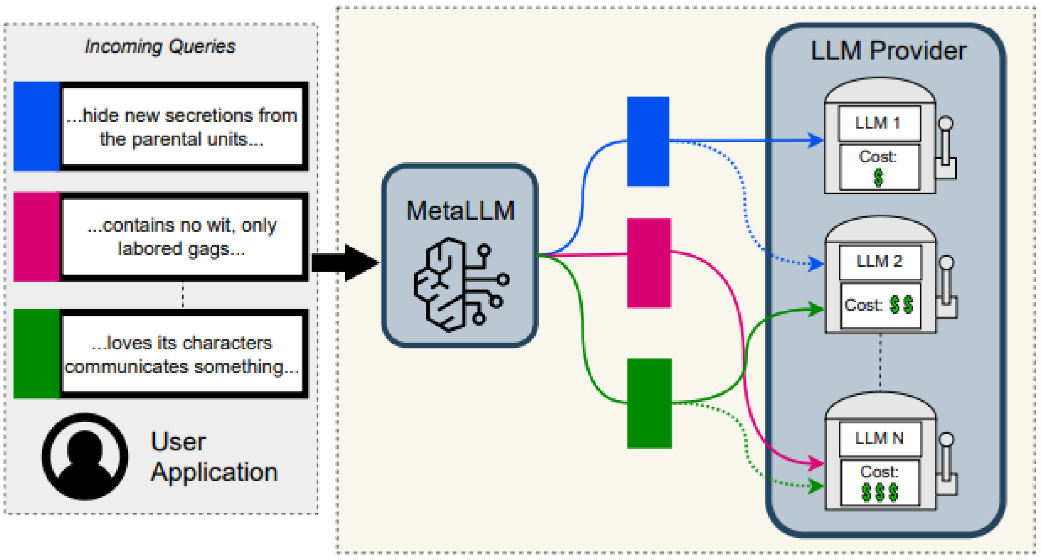

Incoming Queries
...hide new secretions from the parental units...
...contains no wit, only labored gags...
...loves its characters communicates something...
User Application
MetaLLM
LLM Provider
LLM 1
Cost: $
LLM 2
Cost: $$
LLM N
Cost: $$$

Figure 2.2.5 shows how MetaLLM serves queries to LLM Provider

MetaLLM, introduced by [9],frames LLM routing as a multi-armed bandit problem, offering a principled alternative to heuristic-based approaches by balancing prediction accuracy with cost efficiency under uncertainty. Instead of relying on query difficulty estimation, the framework leverages Sentence-BERT embeddings and a linear reward model to guide routing decisions.

Experimental results across commercial APIs demonstrate its effectiveness: on OpenAI models, MetaLLM improves accuracy by around 1% over the best single model while reducing costs by up to 60%, and on Amazon Bedrock APIs it achieves up to 40% cost reduction without compromising performance. By dynamically exploiting the heterogeneous strengths of different LLMs, MetaLLM successfully routes simpler queries to cheaper models and reserves more powerful models for complex tasks, ensuring a robust balance between quality and efficiency.

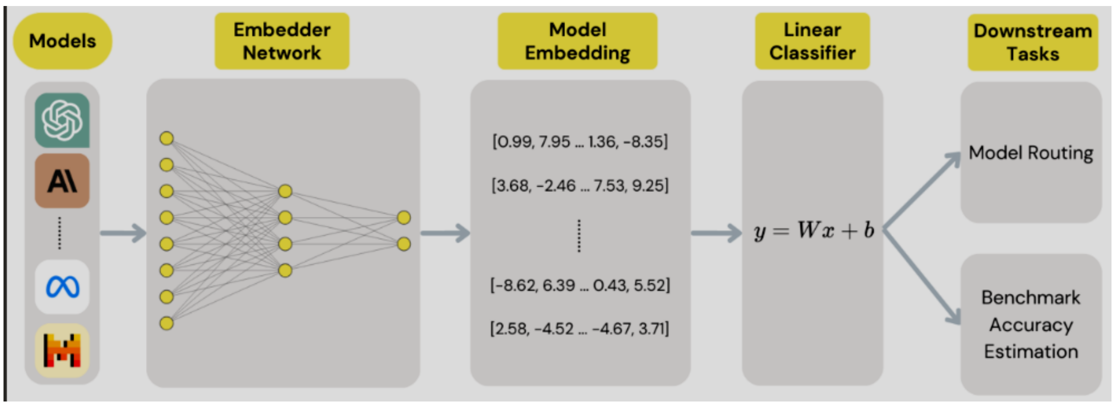


Figure 2.2.5 EmbedLLM proposed pipeline

EmbedLLM, proposed by [8], introduces a framework for representing entire large language models (LLMs) as compact vector embeddings, enabling efficient support for downstream tasks such as routing, correctness prediction, and benchmark evaluation. This approach addresses the inefficiencies of task-specific training by providing a unified, low-dimensional representation that captures salient model characteristics.

Through an encoder–decoder design, EmbedLLM learns embeddings that preserve key performance-related features, allowing simple linear classifiers to be applied for tasks like routing and performance forecasting. Empirical results show that EmbedLLM outperforms prior methods in routing accuracy and latency, while its learned embeddings naturally cluster models with similar capabilities, such as coding-specialized models, even without explicit supervision. Probing experiments further confirm its adaptability, as embeddings dynamically reflect the inclusion of new benchmark tasks.

By enabling accurate performance forecasting across benchmarks without additional inference costs, EmbedLLM provides a scalable and cost-effective solution for managing the rapidly growing ecosystem of LLMs.

Routers	**Routing Formulation**	**Training Signal**	**Training Objective**
TensorOpera	Classification	BERTsim score	Cross Entropy
MetaLLM	Multi-armed Bandit	Online utility score	Linear reward model
RouteLLM	Ranking	Pairwise preference	Contrastive loss
Eagle	Ranking	Pairwise preference	ELO calculation
EmbedLLM	Embedding Similarity	Model representation	Contrastive learning

Table 2.2.1 Summary and comparison of related works

A review of the current literature on LLM routing reveals a strong inclination towards complex training objectives, particularly those based on ranking and relative preference. Prominent works such as RouteLLM and Eagle employ pairwise preference learning with contrastive loss or ELO-based calculations, framing the routing decision as a ranking problem. Similarly, methods like EmbedLLM utilize contrastive learning on model representations, while MetaLLM opts for an online multi-armed bandit approach.

While these sophisticated methods have shown promise, a discernible research gap exists in the application of more direct supervised learning paradigms for this task. The literature has not extensively explored framing the routing challenge as a standard classification problem, where the goal is to assign a prompt to a discrete class based on its embedding. Although TensorOpera uses a classification objective, it relies on a specific BERTsim score as its training signal rather than a direct quality label.

This observation inspired the experimental design of this research project. It seeks to investigate whether the performance achieved by complex ranking-based objectives can be effectively met or approximated by well-established supervised classification algorithms. By implementing and rigorously evaluating models like XGBoost, Logistic Regression, and KNN within the standardized evaluation framework established by the RouteLLM paper, this project directly addresses this gap. The core objective is to determine if a simpler, more direct classification-based approach can serve as a viable and potentially more straightforward alternative to the prevailing ranking and preference-based methods for effective LLM routing.

2.3 Generative AI Benchmarks

Evaluating the capabilities of Large Language Models (LLMs) is necessary to track progress, select the appropriate models for a specific application, and understand their strengths and weaknesses. This section will highlight some of the benchmarks. For measuring model's understanding across subjects in diverse academic disciplines, the Massive Multitask Language Understanding (MMLU) [12] benchmark contains a large dataset of multiple-choice questions across 57 subjects. The scoring of the benchmark is through the percentage of questions answered correctly. Grade School Math 8K (GSM8K) benchmark is a dataset designed to evaluate the multi-step mathematical reasoning ability of the LLMs. The datasets contain questions and answers, and the accuracy of the models are evaluated based on the final answer. Beyond knowledge and reasoning, assessing conversational and instruction-following abilities is crucial. The MT-Bench[13] benchmark is designed for this purpose, testing an LLM's capabilities in multi-turn conversations which better reflect real-world human interactions. MT-Bench consists of 80 multi-turn questions across eight common categories such as writing, reasoning, and coding. Evaluation is automated through an "LLM-as-a-judge" approach, where a powerful model like GPT-4 scores responses on a scale of 1 to 10. This method, which shows high correlation with human preferences, allows for a scalable way to assess crucial attributes like coherence, nuanced reasoning, and alignment with user intent.

CHAPTER 3 – System Model

3.1 Problem Formulation

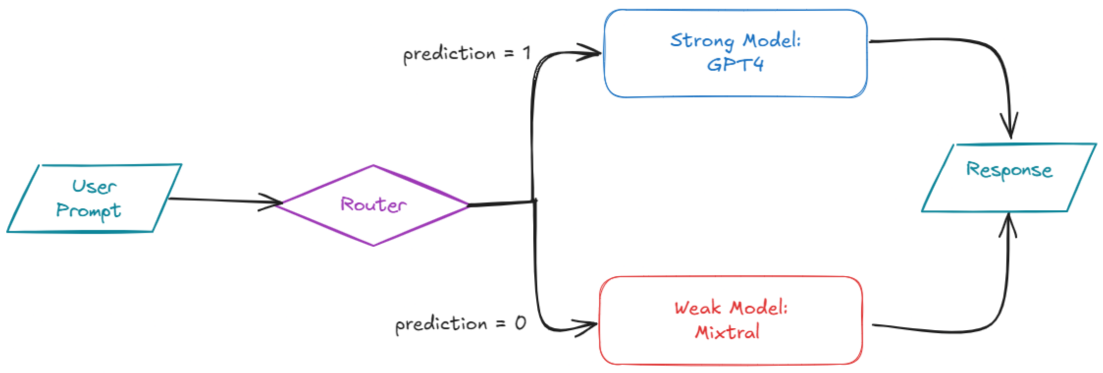


Figure 3.1.1: Overview of the objective of LLM routing

Objective: To design and implement a routing system that intelligently directs user prompts to one of two Large Language Models (LLMs): a highly capable but expensive "strong" model (e.g., GPT-4) or a less capable but cost-effective "weak" model (e.g., Mixtral-8x7B) as shown in figure 3.1.1.

Problem as Classification: The task as a binary classification problem. Given a user prompt, the system must predict a routing label: 1 for the strong model or 0 for the weak model. The prediction should be probabilistic, providing a confidence score used to prioritize which prompts are most likely to benefit from the strong model. Crucially, this prediction is probabilistic. The router outputs a continuous confidence score (from 0.0 to 1.0) representing the likelihood that a prompt requires the strong model. This score is fundamental to our evaluation framework, as it allows us to create a prioritized queue of prompts, enabling a nuanced analysis of performance across various budget constraints. Additionally, the router is train for binary classification to align closely with RouteLLM [5] experimental methodology so the same performance metrics can be used for benchmarking.

Result Evaluation by benchmarking: To ensure the validity and comparability of our results, our experimental methodology is designed to align closely with the framework established in the RouteLLM [5] paper. This alignment is maintained in two critical areas:

1. **Evaluation Datasets**: We utilize the same test datasets, including MT-Bench, GSM8K and the MMLU dataset. This allows for a direct comparison of our router's performance on tasks with diverse characteristics and demonstrates its generalization capabilities.

2. **Performance Metrics**: Our evaluation employs the same core metrics used to measure router efficiency. These include the Cost-Performance Trade-off (CPT) analysis, and Average Performance Gap Recovered (APGR). By adopting these standardized metrics, we can benchmark our system's effectiveness against established results and provide a consistent, meaningful measure of its ability to intelligently manage LLM resources.

3.2 Routing Framework Overview

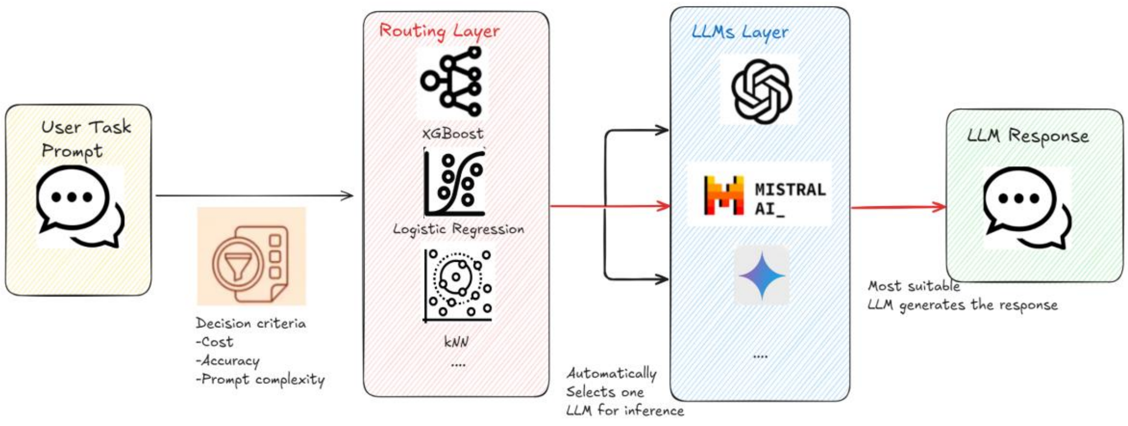


Figure 3.2.1 Overview of Routing Framework

The primary goal is to create routing framework like in Figure 3.2.1 that introduces a dedicated abstraction layer (routing layer) between the user prompt and the pool of available large language models (LLMs). Instead of sending every prompt to a single model (which may be expensive, underpowered, or inconsistent), the routing layer powered by a trained LLM router (e.g., Logistic Regression, XGBoost, Hybrid approaches) analyzes the prompt and predicts which LLM is the most efficient choice. The decision is made based on multiple criteria such as prompt complexity, accuracy requirements, and inference cost. By doing so, the framework ensures that simple prompts can be routed to cheaper, lightweight models, while complex prompts are directed to more capable but expensive models. This abstraction makes the system scalable, cost-efficient, and adaptive, enabling automatic selection of the best LLM without requiring users to manually choose, ultimately improving both performance efficiency and resource optimization in real-world deployments Although for this research, the problem is framed as binary classification problem for the purpose of adhering the benchmarking standards by RouteLLM paper [5], the models in the router layer can be modified for multi-class classification for automatically selecting from more than two LLMs in the LLM selection layer.

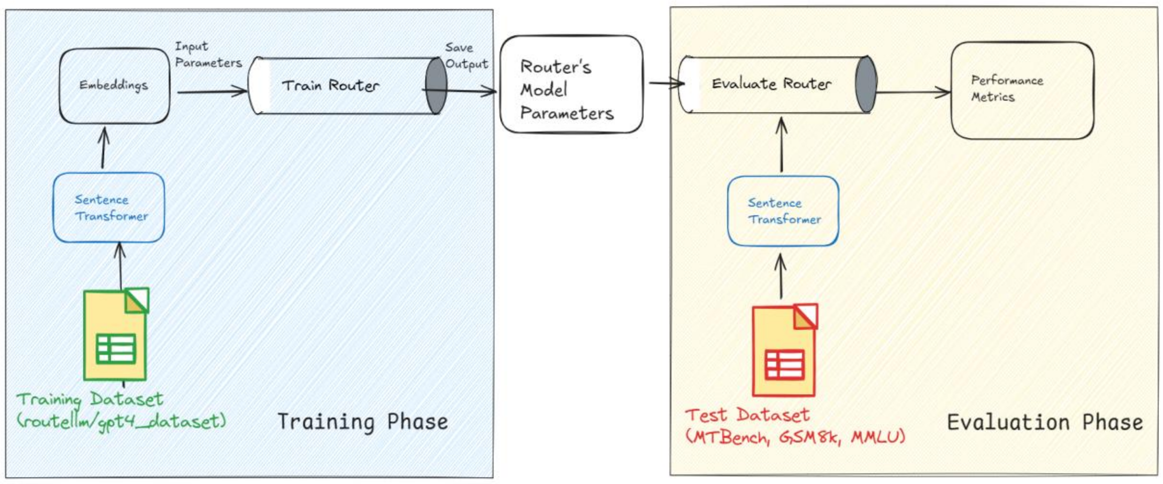


Figure 3.2.2 Training methodology of LLM Routers

As mentioned in the overall framework, the LLM routers needed to be trained and evaluated before deployment. The figure above shows the high-level description of the training methodology for the four different routing model. The process is divided into two distinct stages: a **Training Phase** and an **Evaluation Phase**. This separation ensures that the router is trained on a dedicated dataset and then tested on entirely unseen data, providing unbiased assessment of its performance.

3.2.1 The Training Phase

The objective of the Training Phase is to build and save an intelligent routing model capable of distinguishing between prompts that require a strong LLM and those that can be handled by a weak one. The workflow proceeds as follows:

Training Dataset: The process begins with the routellm/gpt4_dataset. This dataset contains a large number of prompts, each associated with a quality score between 1 to 5 that serves as the basis for generating our ground-truth labels.

Sentence Transformer: The raw text prompts are fed into a pre-trained Sentence Transformer model. This critical component acts as a feature extractor, converting each text-based prompt into a high-dimensional numerical vector, or "embedding." These embeddings capture the semantic meaning of the prompts, allowing the subsequent model to make decisions based on intent rather than just keywords.

Train Router: The generated embeddings, along with their corresponding binary routing labels (the "Input Parameters"), are used to train the router model. This is the core machine learning step where the model learns the complex patterns that correlate a prompt's semantic fingerprint with its probable need for the strong LLM.

Router's Model Parameters: The final output of this phase is the set of learned router's model parameters. This is the tangible artifact such as a saved model file containing the weights and bias or a FAISS index that encapsulates all the similarity index the router has gained. This saved state is then passed to the next phase.

3.2.2 The Evaluation Phase

The purpose of the Evaluation Phase is to assess the performance of the trained router on new, unseen data, simulating its real-world application. This phase is done offline with the prepared dataset.

Test Dataset: This phase utilizes distinct test datasets, including MTBench, GSM8k, and MMLU. These datasets cover a range of tasks, from general conversational quality to specialised mathematical and multi-domain reasoning, allowing us to test the router's generalisation capabilities.

Sentence Transformer: Like the training phase, the prompts from the Test Dataset are first converted into semantic embeddings using the same Sentence Transformer. This ensures consistency in the data representation between training and evaluation.

Evaluate Router: In this step, the saved router's model parameters from the Training Phase are loaded. The router then uses this learned knowledge to generate a probability score for each embedding from the test set. It is here that the router makes its predictions on data it has never seen before.

Performance Metrics: The predictions generated by the router are compared against the true labels of the test set. This comparison is used to calculate a suite of Performance Metrics, including the Cost-Performance Threshold (CPT), and Average Performance Gap Recovered (APGR). These metrics provide a quantitative and comprehensive measure of the router's efficiency and effectiveness.

3.3 Routing Methods

3.3.1 Sentence Transformer

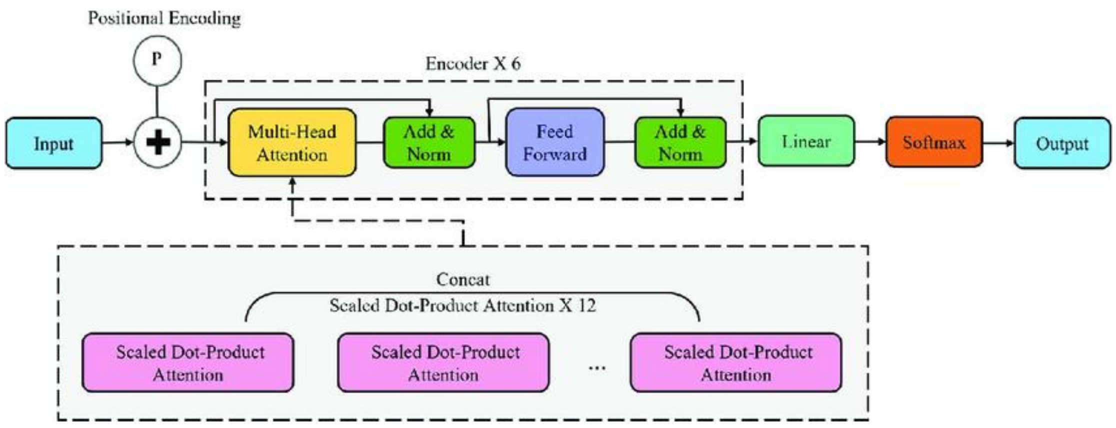

Positional Encoding
P
Encoder X 6
Input
Multi-Head Attention
Add & Norm
Feed Forward
Add & Norm
Linear
Softmax
Output
Concat
Scaled Dot-Product Attention X 12
Scaled Dot-Product Attention
Scaled Dot-Product Attention
...
Scaled Dot-Product Attention

Figure 3.3.1 Architecture of DistilRoBERTa [14]

The model "all-distilroberta-v1" [15] is a sentence embedding model based on DistilRoBERTa-base as shown in figure 3.3.1, fine-tuned with a contrastive learning objective on over 1 billion sentence pairs drawn from many datasets to produce 768-dimensional dense vectors for sentences or short paragraphs. After tokenizing input up to 128 tokens, it uses mean pooling over token embeddings (masking out padded tokens), followed by L2 normalization, so that similar sentences have high cosine similarity.

Before training any classifier, each prompt is encoded into a dense vector representation $x \in R^d$ using a sentence transformer (all-distilroberta-v1) as input to the four routing methods:

$$x = f_{embed}(prompt)$$

where f_{embed} is the pre-trained sentence transformer (all-distilroberta-v1) and d = 768 where every sentence is mapped into a 768-dimensional embedding.

3.3.2 Logistic Regression Classifier

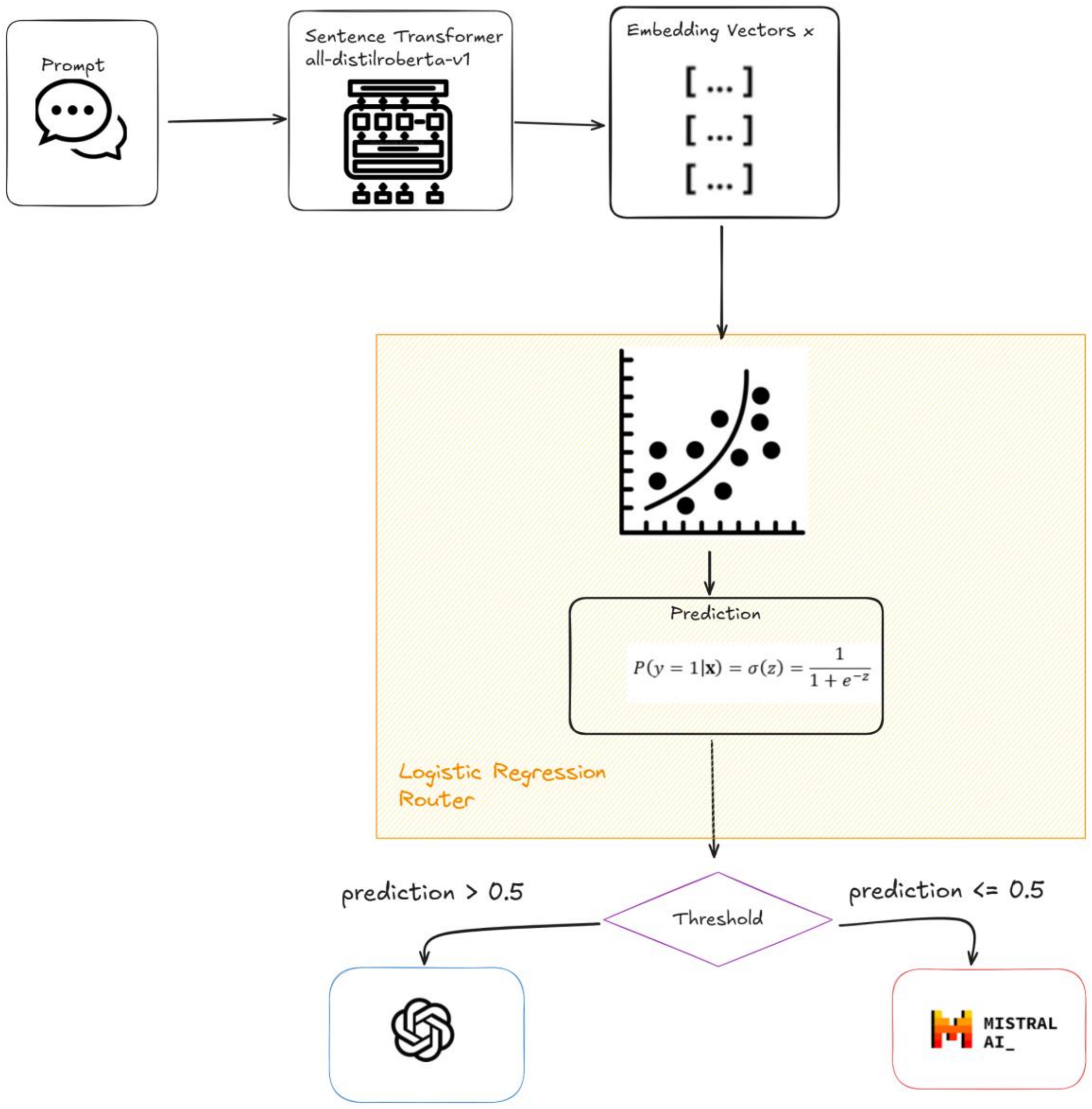


Figure 3.3.2.1 Overview of how Logistic Regression router perform inference

Logistic Regression learns a linear decision boundary in embedding space:

$$z = \mathbf{w}^{\top}\mathbf{x} + b$$

where:

$w \in R^d$ = weight vector,

$b \in R$= bias,

The probability that the input should be routed to the strong model is given by the sigmoid function:

$$P(y = 1|\mathbf{x}) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

The prediction threshold is as follows:

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p} = P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

This mean when the model predicts high probability (>0.5), the model outputs 1 which means route to "strong" LLM model. Otherwise, route to "weak" LLM model.

The model is trained by minimizing the loss using binary cross-entropy:

$$\mathcal{L}(w, b) = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log P(y_i|\mathbf{x}_i) + (1 - y_i)\log(1 - P(y_i|\mathbf{x}_i))]$$

Logistic Regression serves as a simple yet effective baseline for the routing task because of its inherently binary nature (weak vs. strong model). The intuition for this method is that the classifier should learn a linear decision boundary in the embedding space, where prompts are represented as vectors, and applies a sigmoid function to estimate the probability of routing to the strong model. Its simplicity and interpretability make it a natural starting point: if embeddings are well-structured, even this basic model can separate the two classes; otherwise, it highlights the need for more advanced approaches.

3.3.3 XGBoost Classifier

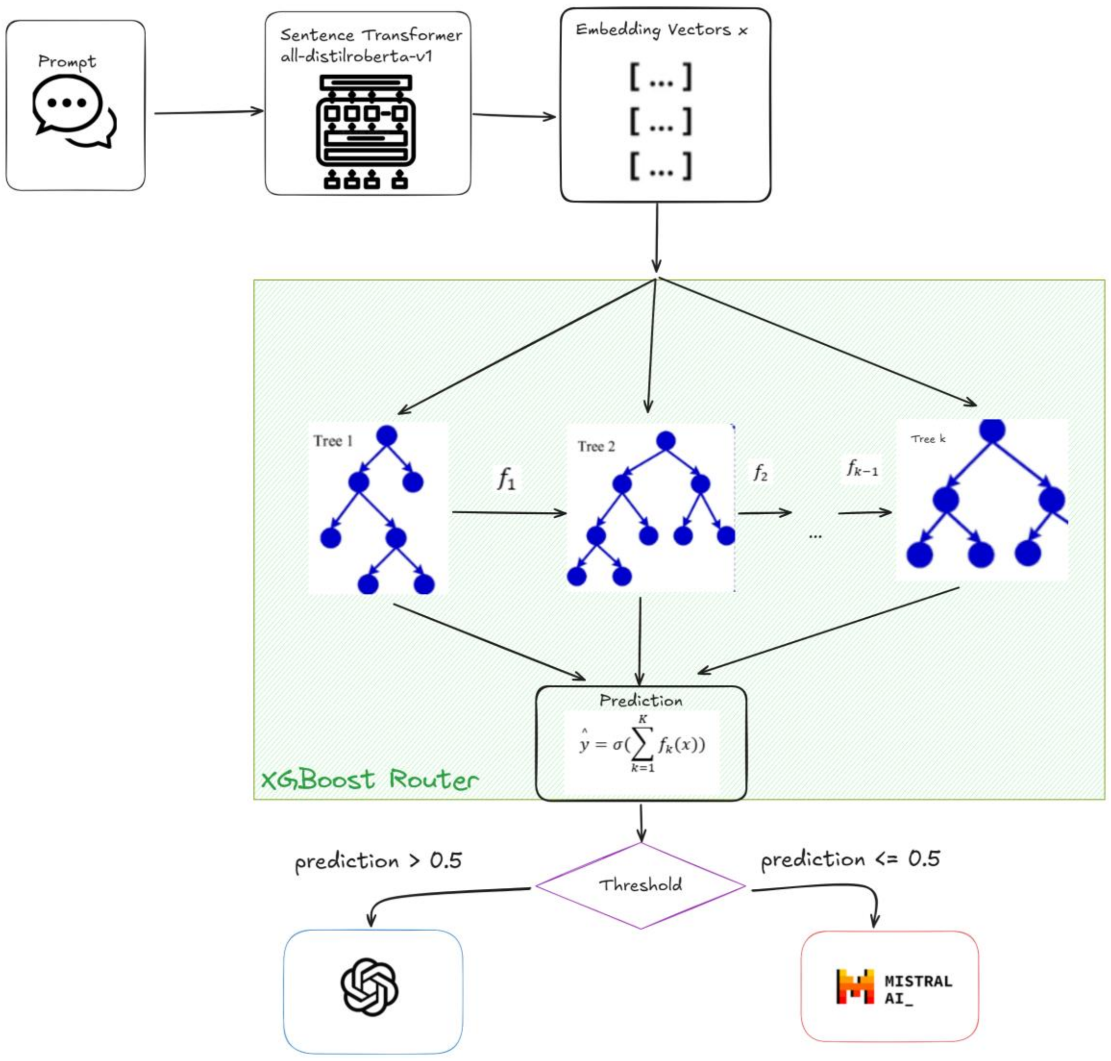


Figure 3.3.3.1 Overview of how XGBoost router perform inference

XGBoost is a gradient boosting algorithm that builds an ensemble of decision trees. Each tree partitions embedding space into regions and outputs a weight with the final logit:

$$\hat{y} = \sigma(\sum_{k=1}^{K} f_k(x))$$

where:

f_k is the k-th decision tree,

K is the number of trees,

The logistic function is $\sigma(z) = \frac{1}{1+e^{-z}}$

The prediction threshold is similar across or routing methods:

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p} = P(y = 1|x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

This mean when the model predicts high probability (>0.5), the model outputs 1 which means route to "strong" LLM model. Otherwise, route to "weak" LLM model.

The loss is calculated using binary cross-entropy:

$$\mathcal{L} = -\sum_{i=1}^{N} (y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i))$$

XGBoost optimizes this with second-order Taylor expansion (gradient+hessian)[16] at each split.

The intuition for this routing method is that the embeddings should capture some semantic difficulty of the prompt which lets XGBoost learns the decision boundaries in the embedding space such that it can separate easy prompts vs hard prompts for routing.

3.3.4 kNN Router

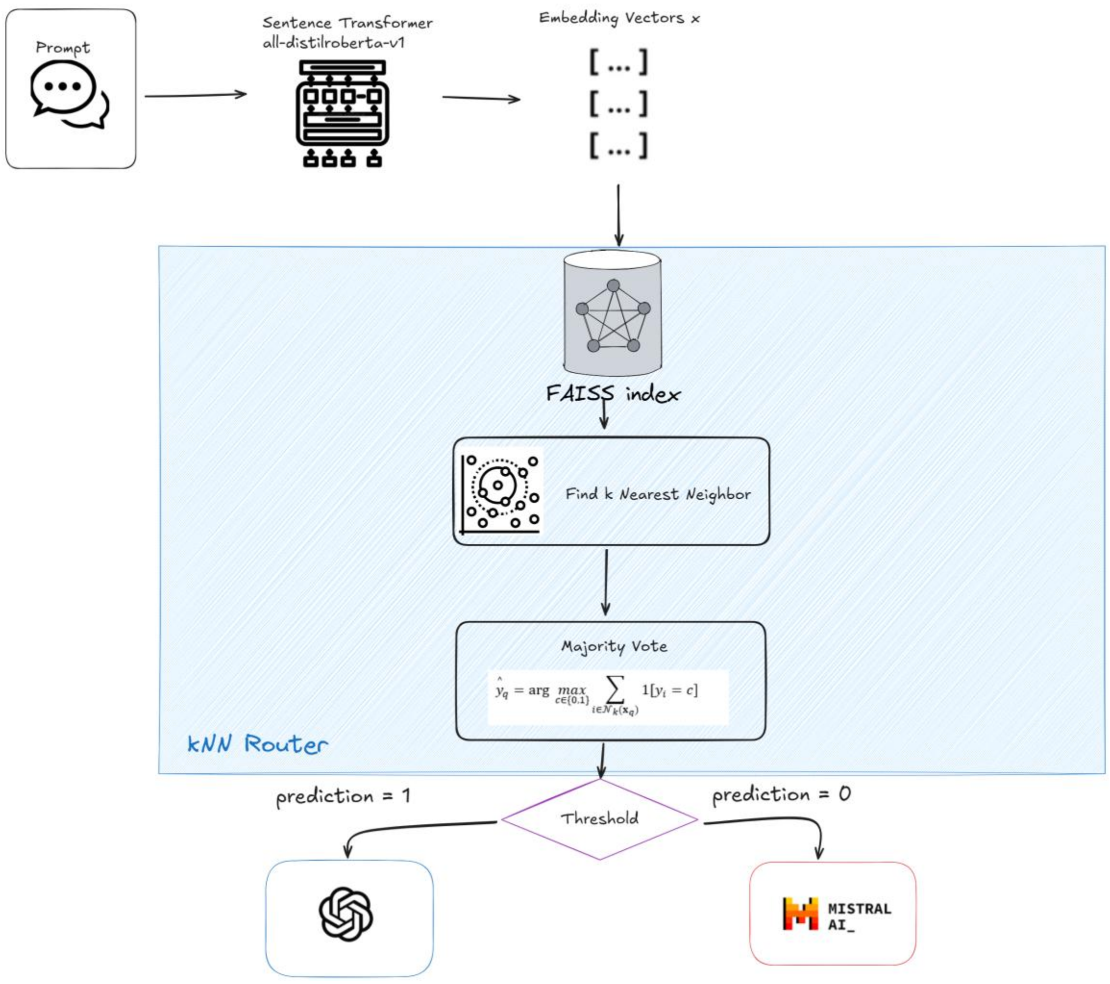


Figure 3.3.4.1 Overview of how kNN router perform inference

FAISS (Facebook AI Similarity Search) is an open-source library developed by Meta (Facebook AI Research) for efficient similarity search and clustering of dense vectors. In this scenario it is used to build an index from the embeddings generated by the sentence transformer.

FAISS works with inner product after normalization to approximate cosine similarity:

$$||x||_2 = 1, \text{sim}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i^{\top}\mathbf{x}_j$$

Since embeddings are L2-normalized, the dot product = cosine similarity.

FAISS build an index of training embeddings as such:

$$\mathcal{I} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$$

where each vector $\mathbf{x}_i$ is associated with a routing label: $y_i \in \{0,1\}$

For a query embedding $\mathbf{x}_q$, FAISS finds the k nearest neighbors:

$$\mathcal{N}_k(\mathbf{x}_q) = \text{arg top-}k\{\text{sim}(\mathbf{x}_q, \mathbf{x}_i) \mid i = 1 \dots N\}$$

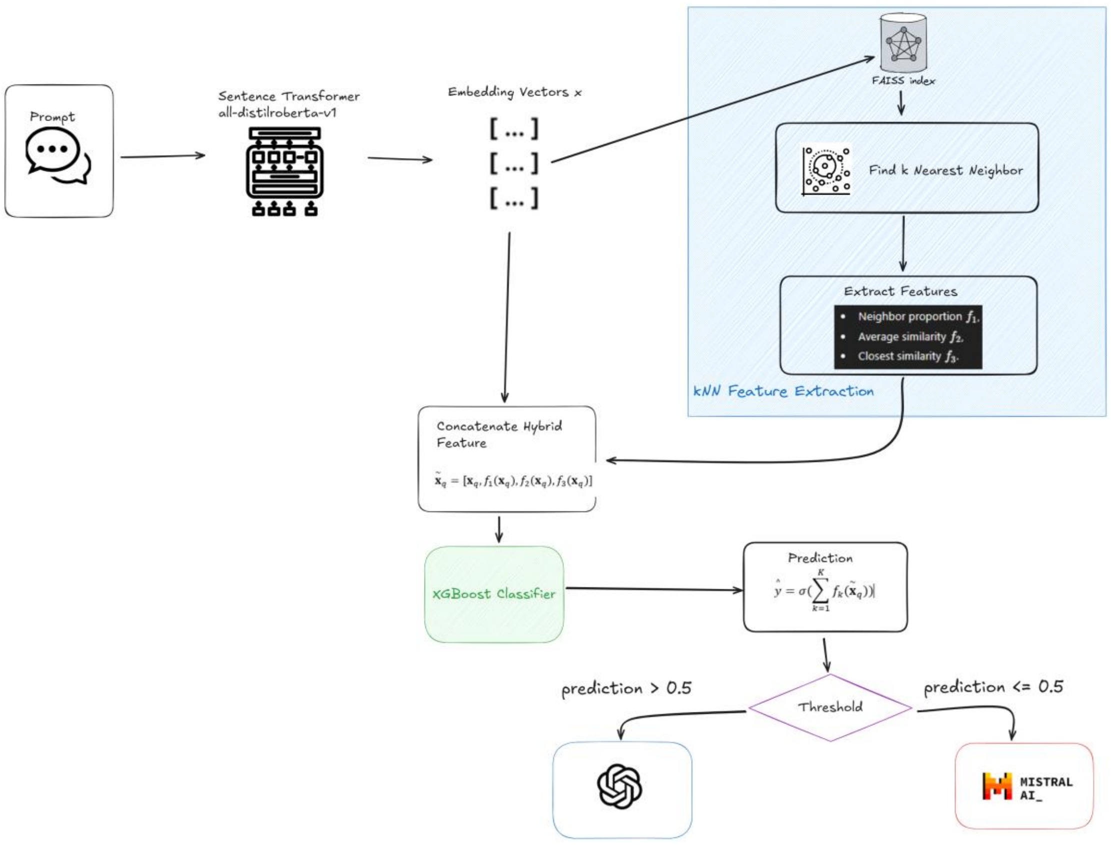
The predicted label is by majority vote:

$$\hat{y}_q = \text{arg} \underset{c\in\{0,1\}}{max} \sum_{i\in\mathcal{N}_k(\mathbf{x}_q)} 1[y_i = c]$$

If the predicted label $\hat{y}_q = 1$, route to strong model, otherwise route to weak model.

The intuition behind this routing approach is that the embeddings should cluster similar prompts. Prompts that were judged "hard" (low Mixtral score → strong model needed) will be near each other in embedding space. The assumption is that a query prompt is likely to need the same routing decision as its neighbors which is why kNN is used to classify the routing decision here. This method requires no training as it simply stores the embeddings and perform similarity search which can be beneficial in some use cases.

3.3.5 Hybrid Approach (kNN + XGBoost Classifier)



Prompt
Sentence Transformer all-distilroberta-v1
Embedding Vectors x
[...]
[...]
[...]
FAISS index
Find k Nearest Neighbor
Extract Features
Neighbor proportion f_1.
Average similarity f_2.
Closest similarity f_3.
kNN Feature Extraction
Concatenate Hybrid Feature
$\tilde{\mathbf{x}}_q = [\mathbf{x}_q, f_1(\mathbf{x}_q), f_2(\mathbf{x}_q), f_3(\mathbf{x}_q)]$
XGBoost Classifier
Prediction
$\hat{y} = \sigma(\sum_{k=1}^{K} f_k(\tilde{\mathbf{x}}_q))$
prediction > 0.5
Threshold
prediction <= 0.5
MISTRAL AI_

Figure 3.3.5.1 Overview of how (kNN+XGBoost) router perform inference

For this approach, a combination of both kNN and XGBoost methods is used to see if more complex embedding features can be extracted and trained to improve performance. In this scenario, the kNN is used as feature extraction to train the XGBoost binary classifier.

The method differs from the other methods in its training set where the training data is split into:

1. Reference set (x_r, y_r) used for similarity search.
2. Meta-training set (x_m, y_m) used to train the classifier.

In this case the FAISS index is built using the reference set:

$$\mathcal{I} = (\mathbf{x}_1, y_1), \dots, (\mathbf{x}_R, y_R)$$

For each query embedding $\mathbf{x}_q$ from the meta-training set or test set:

Get the k nearest neighbours:

$$\mathcal{N}_k(\mathbf{x}_q) = \arg \text{ top-}k\{\mathbf{x}_i^\top \mathbf{x}_q \mid i = 1, \dots, R\}$$

Then extract the following features:

1. Proportion of “strong” label neighbours:

$$f_1(\mathbf{x}_q) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x}_q)} 1[y_i = 1]$$

2. Average similarity to neighbours:

$$f_2(\mathbf{x}_q) = \frac{1}{k} \sum_{i \in \mathcal{N}_k(\mathbf{x}_q)} \text{sim}(\mathbf{x}_q, \mathbf{x}_i)$$

3. Closest neighbour similarity:

$$f_3(\mathbf{x}_q) = \max_{i \in \mathcal{N}_k(\mathbf{x}_q)} \text{sim}(\mathbf{x}_q, \mathbf{x}_i)$$

Afterwards, the features are concatenated with the embeddings:

$$\tilde{\mathbf{x}}_q = [\mathbf{x}_q, f_1(\mathbf{x}_q), f_2(\mathbf{x}_q), f_3(\mathbf{x}_q)]$$

Then, the XGBoost classifier learns from these hybrid features:

$$\hat{y} = \sigma(\sum_{k=1}^{K} f_k(\tilde{\mathbf{x}}_q))$$

And generates the prediction with the following threshold:

$$\hat{y} = \begin{cases} 1 & \text{if } \hat{p} = P(y = 1 | x) > 0.5 \\ 0 & \text{otherwise} \end{cases}$$

The intuition behind this routing method is to use kNN to extract more contextual difficulty cues from prompts such that XGBoost can learn from these features.

3.4 Timeline

Week/ Tasks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**

Write up Chapter 1 - Introduction	🟦	🟦											
Write up Chapter 2 – Literature Review			🟦	🟦									
Propose and Design Solution					🟦	🟦							
Develop proposed solution							🟦	🟦					
Write up Chapter 3 – Proposed Solution									🟦				
Write up Chapter 4- Preliminary work										🟦			
Demo and submit draft to supervisor											🟦		
Final Check and submission of report.												🟦	
Presentation													🟦

Table 3.4.1 Timeline of FYP1

Week/ Tasks	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Title Refinement	🟦	🟦												
Conduct Experiment			🟦	🟦	🟦	🟦								
Benchmark results							🟦	🟦						
Data analysis									🟦	🟦				
Write up report 2											🟦	🟦		
Final Check and submission of report.													🟦	
Presentation														🟥

Table 3.4.2 Timeline of FYP2

For FYP2, title refinement was discussed with supervisor and done in the first two weeks of the June 2024 Trimester. The project scope has been narrowed down and shifted its focus to a single core component that was mentioned in FYP1 which is the LLM selection algorithm now framed as a research into LLM routing within the Gen AI area of study. This required revisiting the literature to adopt established evaluation frameworks, which subsequently drove significant redesigns of the proposed system architecture to ensure the results could be rigorously benchmarked.

CHAPTER 4 – Experiment

4.1 Hardware Setup

Description	Specifications
Processor	13th Gen Intel(R) Core (TM) i5-13420H 2.10 GHz
Operating System	Windows 11
Graphic	Intel® UHD Graphic
Memory	16.0 GB DDR4 RAM
Storage	1TB SDD

Table 4.1.1 Specification of Computer

Description	Specifications
Graphic	NVIDIA Tesla P100
GPU Memory	16GB
Usage	Utilised for training phase

Table 4.1.2 GPU Specifications used for Training Phase

The training and evaluation of the experiment were conducted on the Kaggle platform to leverage its free GPU resources. The system was equipped with an NVIDIA Tesla P100 GPU with 16 GB memory, as summarized in table 4.1.2, which was primarily utilized during the training phase.

4.2 Software Setup

To implement and evaluate the proposed models, the experiment was conducted using Kaggle Notebooks, which provide a cloud-based environment with free access to GPU resources. A range of Python libraries and frameworks were imported within the notebook to support various tasks, including data preprocessing, model training, experiment tracking, visualization, and system interaction. The key components and their purposes are summarized in Table 4.2.1.

Description	Component/Library	Purpose
Platform	Kaggle Notebooks	A cloud-based computational environment for executing code
GPU	NVIDIA Tesla P100	A powerful GPU provided by Kaggle for accelerating machine learning tasks.
Experiment Logging	Weights & Biases (wandb)	For logging metrics, visualizing model performance, and tracking experiments.
Data Processing	datasets	To efficiently load and process datasets.
	pandas	For data manipulation and analysis, particularly with tabular data.
	numpy	For fundamental numerical computations.
Machine Learning Models	xgboost	A gradient boosting library for building high-performance models.
	sentence-transformers	For creating embeddings from sentences and text.
	faiss	For efficient similarity search and clustering of dense vectors
	scikit-learn	Used here for splitting data (train_test_split) and evaluating model performance (accuracy_score).
Data Visualisation	matplotlib.pyplot	For creating static, animated, and interactive visualizations.
System Interaction	os	To interact with the operating system, such as managing file paths.

Table 4.2.1: Libraries and Components Used in the Experimental Setup

4.3 Setting and Configuration

To ensure fair evaluation and reproducibility across experiments, each machine learning model was configured with carefully selected hyperparameters. These configurations control the training dynamics, optimization process, and model complexity. The following tables summarize the key hyperparameters for the models used in this study: Logistic Regression (Table 4.3.1), XGBoost (Table 4.3.2), and k-Nearest Neighbors (k-NN) (Table 4.3.3).

Model Hyperparameters	**Value**	**Purpose**
random_state	42	Ensures reproducibility of results by fixing the randomness in data shuffling and weight initialization.
max_iter	1000	Sets the maximum number of iterations the solver will run, allowing convergence when the dataset or optimization problem is complex.

Table 4.3.1: Hyperparameters of Logistic Regression Model

Model Hyperparameters	**Value**	**Purpose**
objective	binary:logistic	Defines the task as a binary classification problem with a probabilistic output.
eval_metric	logloss	The metric used on the validation set to monitor for early stopping.
n_estimators	1000	The maximum number of decision trees to build in the ensemble.
max_depth	6	The maximum depth of any individual tree, controlling model complexity.
learning_rate	0.05	A small step size (eta) that reduces the influence of each tree, preventing overfitting.
subsample	0.8	The fraction of the training data randomly sampled for growing each tree.
colsample_bytree	0.8	The fraction of features (embedding dimensions) randomly sampled for each tree.
early_stopping_rounds	50	The training process will halt if the logloss on the validation set does not

		improve for 50 consecutive rounds, ensuring the optimal number of trees is used.

Table 4.3.1: Hyperparameters of XGBoost Model

Model Hyperparameters	**Value**	**Purpose**
n_neighbors (k)	10	The number of nearest neighbors to retrieve from the reference set for the majority vote.
metric	cosine similarity	The distance metric used to measure similarity between prompt embeddings. Cosine similarity is well-suited for high-dimensional semantic vectors.

Table 4.3.1: Hyperparameters of KNN

4.4 Experimental Setup

4.4.1 Datasets

Training data: routellm/gpt4_dataset [17], a publicly available dataset hosted on the Hugging Face Hub is used for training our machine learning model. The dataset contains 109,101 rows of user's input query. Data augmentation was already performed on this dataset to generate the label for each query using LLM as a Judge technique by the provider. Hence, there is no need of spending additional API cost from GPT-4 to label the dataset. This dataset is specifically designed for tasks related to language model evaluation and is chosen to save the cost of labelling the data.

It consists of two primary columns relevant to our task:

- prompt: The raw text of the user's input query.
- mixtral_score: A numerical score, judged by GPT-4, that rates the quality of the response generated by the "weak" model (Mixtral-8x7B) for the corresponding prompt on a scale of (1-5).

As recommended by [18], the prompt with mixtral_score above 4 shows that Mixtral-8x7B produces strong answer and should be labelled “0” to route to “weak” model. Otherwise, label other prompts with mixtral_score below 4 to route to “strong” model with the label “1”. The training dataset is split 80% for training and 20% for validation. The purpose of the validation set is to provide an unbiased evaluation of the model’s performance on unseen data to monitor for overfitting and understand how well the model is generalising. Additionally, the random seed is set to 42 to ensure the split is deterministic.

Evaluation Benchmarks: The effectiveness of the developed routing system is benchmarked against three diverse and widely-recognized evaluation datasets similar to the RouteLLM [5] paper as mentioned in section 3.1 . This selection provides a comprehensive assessment of the router's ability to generalize across conversational, knowledge-based, and reasoning-intensive tasks. The specific benchmarks are:

- MMLU [12]: A large-scale, multi-task benchmark designed to measure knowledge across 57 academic subjects, comprising 14,042 questions.
- MT-Bench [13]: A multi-turn conversational benchmark with 160 qualitative prompts, where an LLM-as-a-judge methodology is used for scoring.
- GSM8K [19]: A dataset of 1,319 rows of elementary-level mathematical reasoning problems designed to test quantitative and logical skills.

MT-Bench contains judge score of (1-10) for Mixtral-8x7B, which is labelled “0” to route to “weak” model if score is above 8 and route to “strong” model if below. The MMLU and GSM8K datasets contains (true/false) labels for the strong and weak models’ responses.

4.4.2 Performance Metrics

To analyze the effectiveness of the router model, a series of evaluation metrics are employed which are obtained from the RouteLLM paper [5]. These metrics are designed to capture the inherent trade-off between the computational cost of routing decisions and the resulting performance. The evaluation begins by assessing cost and performance independently, followed by two composite metrics that provide a holistic view of the router's efficiency.

First, we defined some key variables for the evaluation metrics:

- Weak Model (M_w): A faster, cheaper, but less capable language model.
- Strong Model (M_s): A slower, more expensive, but more powerful and accurate language model.
- Router Model (M_{R^α}): An intelligent system that decides for each incoming query whether to send it to the weak model (to save money) or the strong model (for better quality). The α represents a setting or threshold within the router that adjusts its sensitivity for sending queries to the strong model.

To calculate the cost efficiency, the formula below is defined:

$$c(M_{R^\alpha}) = \frac{1}{|Q|}\sum_{q \in Q} \mathbb{I}\{R^\alpha(q) = M_\mathrm{s}\}$$

This formula calculates the percentage of queries that were sent to the strong model by going through every query (q) in a test set (Q). For each query, an indicator function (𝕀) checks if the router (Rα) sent it to the strong model (M_s). If it did, the function outputs a 1; otherwise, it outputs a 0. It sums up all these 1s and 0s and then divides by the total number of queries (|Q|). To summarise, if this value is 0.25, it means the router sent 25% of all queries to the expensive strong model, resulting in significant cost savings compared to using the strong model 100% of the time.

$$r(M_{R^\alpha}) = \frac{1}{|Q|}\sum_{q \in Q} s(M_{R^\alpha}(q))$$

The formula above calculates the average quality score across all answers. For each query (q), the system gets a quality score (s(...)). This score could be a correctness rating from an automated test (like MMLU), a quality rating from a human evaluator (e.g., on a scale of 1 to 10) or rated through LLM-as-a-Judge rating. The formula adds up these scores for all queries and divides by the total number of queries (|Q|).

$$\mathrm{PGR}(M_{R^\alpha}) = \frac{r(M_{R^\alpha}) - r(M_w)}{r(M_s) - r(M_w)}$$

The formula above calculates the Performance Gap Recovered (PGR) of the router. It measures how effectively the router closes the performance gap between the weak and strong models. It contextualizes the router's performance, showing how much of the potential quality improvement it achieved. For example, a PGR of 0.8 means the router has successfully closed 80% of the performance gap, giving you 80% of the benefit of the strong model but at a potentially much lower cost. A PGR of 1.0 (or 100%) means the router performs just as well as using the strong model exclusively.

Conceptual Equation:

$$\mathrm{APGR}(M_{R^{\alpha}}) = \int_0^1 \mathrm{PGR}(M_{R^{\alpha}})d(c(M_{R^{\alpha}}))$$

Practical Equation:

$$\mathrm{APGR}(M_{R^{\alpha}}) \approx \frac{1}{10}\sum_{i=1}^{10} \mathrm{PGR}(M_{R^{\alpha_i}})$$

A good router should not only perform well at high costs but also provide significant improvements at low costs. The formulas above (APGR), measures this overall capability.

Conceptually, it's the "area under the curve" of a graph that plots performance (PGR) against cost (c($M_{R^{\alpha}}$)). A larger area means the router is more efficient across all budget levels. In practice, it is calculated by testing the router at 10 different cost points (e.g., sending 10%, 20%, 30%... up to 100% of queries to the strong model). The PGR is measured at each of these 10 points and then simply take the average.

Additionally, another important metric is Call-Performance Threshold (CPT). It answers the question of "To achieve a certain quality target, what is the minimum cost we must pay?". A CPT(x%) represents the minimum percentage of calls that must be routed to the strong model to achieve a Performance Gap Recovered (PGR) of at least x%. For example, CPT(50%) ≈ 40% means that to close at least 50% of the performance gap between the weak and strong models, you absolutely must send a minimum of 40% of your queries to the strong model. Any lower percentage will not meet the desired quality bar.

4.5 Implementation Issues and Challenges

During the development of this project, several significant challenges were encountered. These ranged from initial data acquisition and labelling to the establishment of an evaluation framework.

1. Data Labeling and Associated Costs

A primary challenge was the acquisition of a suitably labeled training dataset. The initial plan was to manually label a dataset by processing user prompts through both a "weak" and a "strong" Large Language Model (LLM). Subsequently, a powerful judge model, such as GPT-4, would be used to score the responses and assign routing labels. This approach, known as "LLM-as-a-judge," would have provided a high degree of control over the data quality.

However, a preliminary cost analysis revealed that this undertaking would be financially prohibitive. Labelling a large dataset, such as one with over 100,000 entries, would require a substantial number of API calls to the OpenAI platform. The associated costs for generating responses and then using a sophisticated model like GPT-4 for evaluation would quickly escalate, making this approach impractical within the project's budget.

This led to a significant effort to find a suitable, publicly available dataset that was already labeled for a similar routing task. The discovery of the routellm/gpt4_dataset [17] on the Hugging Face Hub was a crucial turning point. This dataset, which had been augmented and labeled using the very LLM-as-a-judge technique we had initially considered, allowed us to bypass the costly labeling phase and proceed with model training.

2. Establishing Standardized Performance Metrics

A second major challenge was the lack of standardized performance metrics for LLM routing systems. As a relatively new and niche area of study, there is no broad consensus on a definitive set of metrics to evaluate the effectiveness of a model that routes prompt between different LLMs. This made it difficult to benchmark our model's performance against other existing or future systems in a standardized way.

Our research for a suitable evaluation framework led us to the RouteLLM paper. This work provided a structured and appropriate evaluation methodology, which we adopted for our project. The paper's framework offered a clear path to assessing the performance of our router, allowing for a more rigorous and replicable evaluation.

3. Reverse Engineering Evaluation Metrics

While the RouteLLM paper provided a valuable framework, a further challenge arose in the practical implementation of its evaluation metrics. The process of reverse-engineering the exact code and methodology for computing these metrics from the paper was a complex and time-consuming task. It required a deep understanding of the benchmarks used, such as MMLU, MT-Bench, and GSM8K, and careful implementation to ensure our results would be comparable. This analysis of the paper's descriptions, the author’s open-sourced repositories and, in some cases, making reasoned assumptions to replicate their reported evaluation techniques accurately.

CHAPTER 5 – Experiment Evaluation and Discussion

5.1 Results

The graphs in Figures 5.1.1, 5.1.2, and 5.1.3 visually represent the trade-off between accuracy and cost for each benchmark. To interpret the graph, each figure plots the resulting accuracy (y-axis) against the percentage of queries routed to the strong model (x-axis), which represents the operational cost. The gray dashed line indicates the performance of a random a baseline and any effective router must have a curve that sits above this line. An ideal router is one that achieves the highest accuracy with the fewest calls to the strong model, meaning its curve should be positioned as far to the top-left of the graph as possible.

The accompanying Tables 5.1.1, 5.1.2, and 5.1.3 provide a quantitative summary of each router's performance using two key metrics: Average Performance Gap Recovered (APGR) and Call-Performance Threshold (CPT).

- **Average Performance Gap Recovered (APGR):** This metric measures the overall efficiency of a router across all cost levels with up to precision of 4 significant figures. Conceptually, it represents the area under the performance curve shown in the figure, with a higher APGR value signifying a more effective router.
- **Call-Performance Threshold (CPT(x%)):** This metric answers the question, "What is the minimum cost required to achieve a certain quality target?" A lower CPT is better, as it indicates the router can reach the desired performance with fewer calls to the expensive strong model.
- **Improvement:** The final column quantifies the percentage gain in APGR over the random baseline, making it easy to compare the relative effectiveness of each routing strategy.

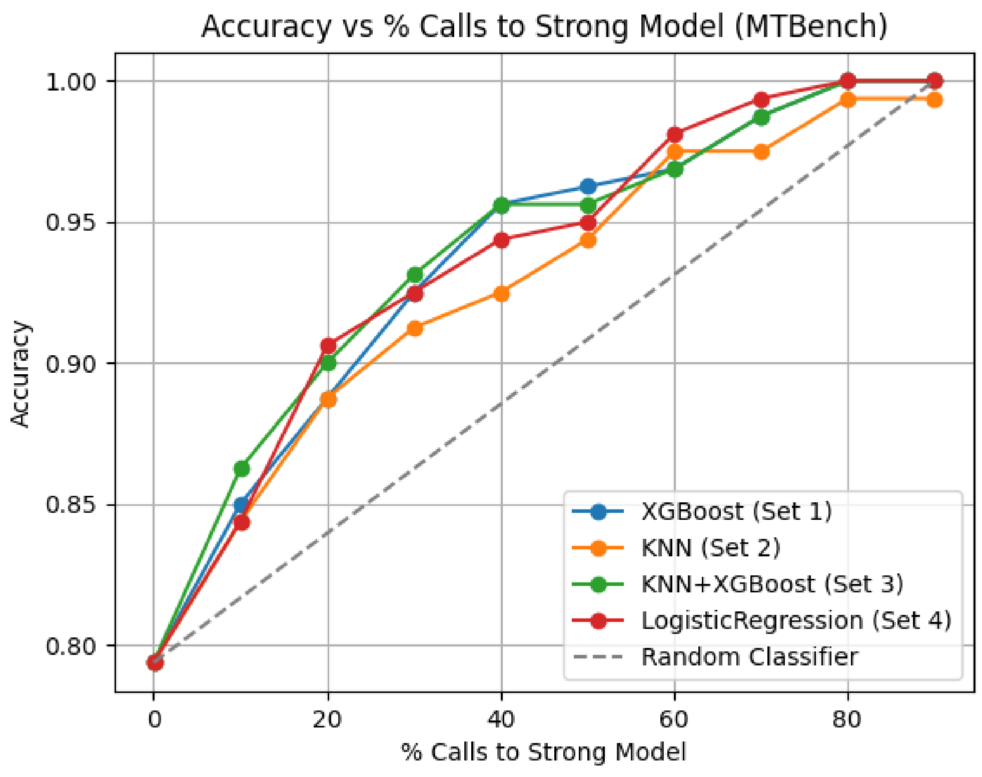


Figure 5.1.1: The routers trade-off between accuracy and cost for MT-Bench Benchmark

Method	CPT(50%)	CPT(80%)	APGR	Improvement
Random (95%CI)	49.03(±4)%	78.08(±3)%	0.500(±0.02)	(+0%)
LogisticRegression	18.75	53.75	0.699	+39.80%
XGBoost	21.88	40.00	0.695	+39.00%
KNN	21.25	52.50	0.650	+30.00%
KNN+XGBoost	**17.50**	**39.38**	**0.709**	**+41.80%**

Table 5.1.1: MT-Bench benchmark results

The evaluation on the MT-Bench dataset reveals substantial efficiency gains from intelligent routing, with all implemented models significantly outperforming the Random baseline's APGR of 0.500 like in [17] . While the supervised classifiers like

XGBoost and Logistic Regression (APGR ≈ 0.69-0.70) proved more effective than the purely similarity-based KNN approach (APGR 0.650), the hybrid KNN+XGBoost method emerged as the definitive top performer. This model achieved the highest overall routing efficiency with an APGR of 0.709, representing a 41.80% improvement over the baseline. Critically, it was also the most cost-effective, requiring the fewest calls to the strong model to achieve both the 50% and 80% performance gain thresholds, with CPT scores of 17.50% and 39.38% respectively. This superior performance highlights the benefit of extracting engineered embeddings features derived from its local neighborhood similarity, which provides the final classifier with a more robust signal for making nuanced routing decisions.

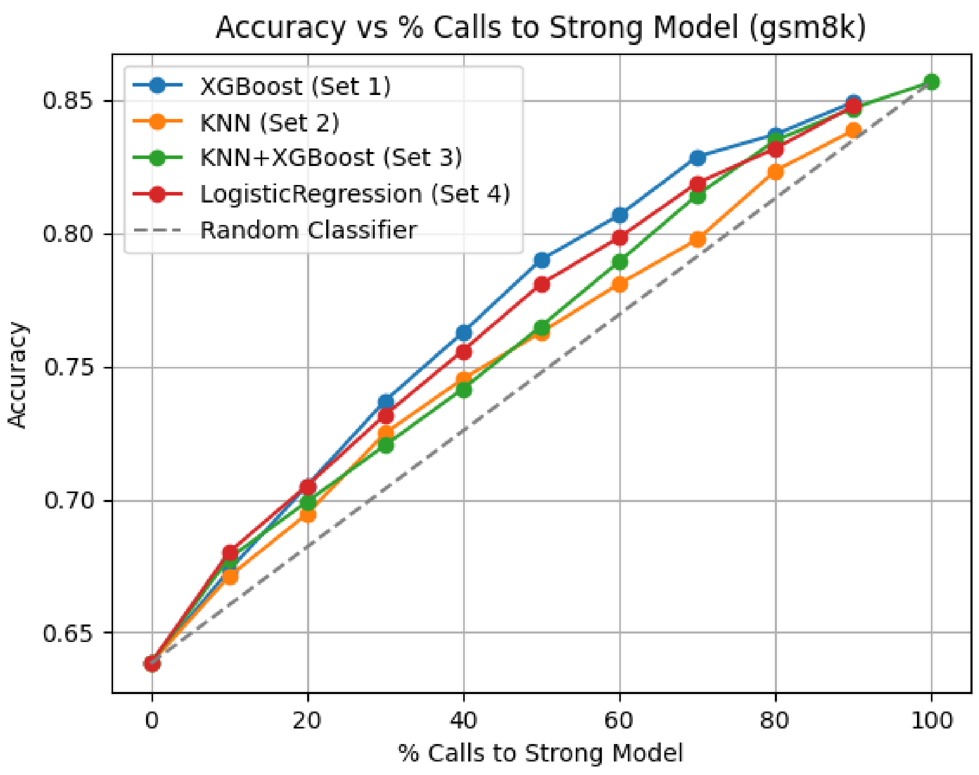
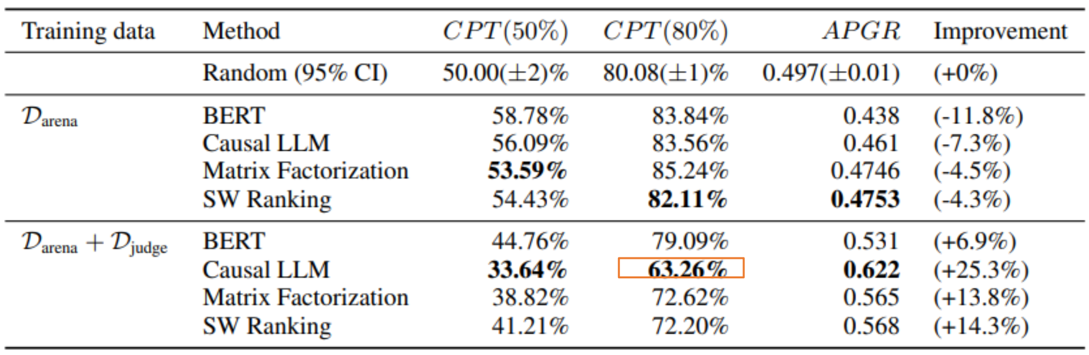


Figure 5.1.2: The routers trade-off between accuracy and cost for GSM8K Benchmark

Method	CPT(50%)	CPT(80%)	APGR	Improvement
Random (95%CI)	50.00(±2)%	80.08(±1)%	0.497(±0.01)	(+0%)
LogisticRegression	35.11	66.57	0.560	+12.68%

XGBoost	34.12	**62.40**	0.579	+16.50%
KNN	42.01	76.27	0.505	+1.61%
KNN+XGBoost	42.76	69.38	0.573	+15.29%

Table 5.1.2: GSM8K benchmark results



Training data	Method	$CPT(50\%)$	$CPT(80\%)$	$APGR$	Improvement
	Random (95% CI)	50.00(±2)%	80.08(±1)%	0.497(±0.01)	(+0%)
$\mathcal{D}_{\text{arena}}$	BERT	58.78%	83.84%	0.438	(-11.8%)
	Causal LLM	56.09%	83.56%	0.461	(-7.3%)
	Matrix Factorization	**53.59%**	85.24%	0.4746	(-4.5%)
	SW Ranking	54.43%	**82.11%**	**0.4753**	(-4.3%)
$\mathcal{D}_{\text{arena}} + \mathcal{D}_{\text{judge}}$	BERT	44.76%	79.09%	0.531	(+6.9%)
	Causal LLM	**33.64%**	**63.26%**	**0.622**	(+25.3%)
	Matrix Factorization	38.82%	72.62%	0.565	(+13.8%)
	SW Ranking	41.21%	72.20%	0.568	(+14.3%)

Figure 5.1.2.1: The GSM8K Benchmark results from RouteLLM [5]

The evaluation on the GSM8K benchmark, which focuses on mathematical reasoning, presented a significantly more challenging routing task compared to the conversational MT-Bench. As shown in the table, the overall performance gains were more modest, indicating that general-purpose semantic embeddings are less effective at distinguishing prompt difficulty in this specialized domain. The purely similarity-based KNN model showed a near-total failure, with an APGR of just 0.505, providing a marginal 1.61% improvement that is barely better than the random baseline. This suggests that raw semantic similarity does not correlate well with mathematical complexity. The supervised models, however, proved more capable, with the standalone XGBoost model emerging as the top performer with an APGR of 0.579, a 16.50% improvement. Notably, the hybrid KNN+XGBoost model saw diminished returns, performing slightly worse than XGBoost alone, likely because the noisy signal from the ineffective KNN component hindered the classifier. A key achievement of this study is the performance of the XGBoost model on the CPT(80%) metric, where it achieved a score of 62.40%. This result is particularly noteworthy as it surpasses the

CPT(80%) performance reported for the best routing method in the RouteLLM paper as shown in the Figure 5.1.2.1 , demonstrating that the implemented model is exceptionally adept at identifying the most complex mathematical problems that necessitate the use of the strong LLM.

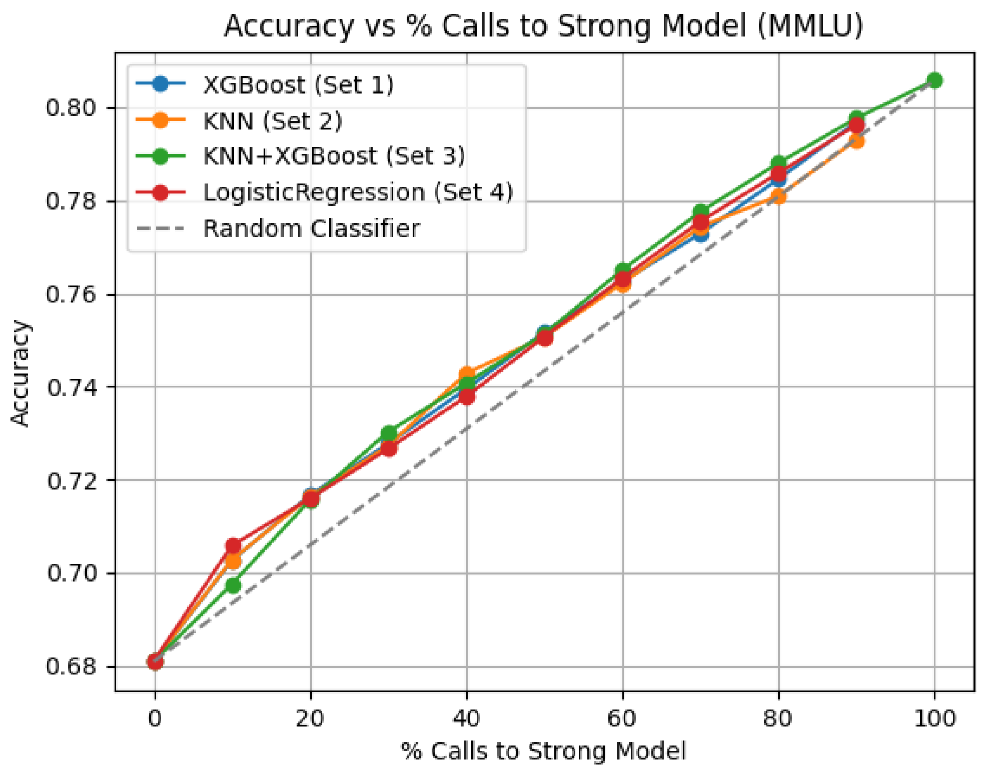


Figure 5.1.3: The routers trade-off between accuracy and cost for MMLU Benchmark

Method	CPT(50%)	CPT(80%)	APGR	Improvement
Random (95%CI)	50.07(±0)%	79.93(±0)%	0.500(±0)	(+0%)
LogisticRegression	44.14	74.40	0.509	+1.80%
XGBoost	42.73	77.07	0.484	-3.20%
KNN	43.80	76.27	0.504	+0.80%
KNN+XGBoost	41.46	74.18	0.557	+11.4%

Table 5.1.3: MMLU benchmark results

The MMLU benchmark, with its vast scope across 57 diverse academic subjects, proved to be the most formidable routing challenge, exposing the limitations of relying solely on general-purpose semantic embeddings. The results demonstrate a near-complete failure of the standard models, with Logistic Regression and KNN achieving negligible improvements over the random baseline (1.80% and 0.80% respectively). Strikingly, the standalone XGBoost model performed worse than random, yielding a negative improvement of -3.20% (APGR 0.484). This suggests that the model likely overfitted to non-generalizable patterns in the training data, as the semantic embedding alone does not provide a reliable signal for difficulty across such a wide array of specialized topics. In this challenging context, the hybrid KNN+XGBoost model was the only architecture to achieve a meaningful performance gain, securing an 11.4% improvement with an APGR of 0.557. This surprising result indicates a powerful synergistic effect: while the individual KNN and XGBoost models failed, the similarity features generated by KNN provided a crucial, stabilizing signal for the XGBoost meta-classifier. This "neighborhood information" appears to be a more robust indicator of prompt difficulty than the raw embedding features alone, allowing the hybrid model to succeed where the others failed and highlighting the necessity of advanced architectures for complex, multi-domain routing tasks.

5.2 Cost Analysis

Based on RouteLLM paper [5], the evaluations are done with gpt-4-1106 API endpoint. For the calculation, they use the pricing for that endpoint [20] which is \$10 per 1 million input tokens and \$30 per 1 million output tokens. To simplify the estimation, the evaluation assumes a single turn setting with short prompts, where the average input length is 95 tokens and the average output length is 264 tokens. This yields an input–output ratio of roughly 95:264. Based on this ratio, the average cost of using GPT-4 can be expressed as

$$\text{Cost}_{\text{GPT-4}} = \frac{95 \times 10 + 264 \times 30}{95 + 264} \approx 24.7 \text{ USD per million tokens.}$$

In the paper, it is said that for Mixtral 8×7B, input and output tokens are priced equally, resulting in an average cost of $0.24 per million tokens.

Building on these estimates, the cost-saving ratios at CPT(50%) and CPT(80%) are defined as the inverse of the ratio of GPT-4 calls made by router relative to the random baseline (50% call to GPT-4 for CPT(50%) and 80% call to GPT-4 for CPT(80%). Specifically:

$$\text{SavingRatio}_{50} = \frac{50\% \cdot \text{Mixtral}_{\text{cost}} + 50\% \cdot \text{GPT4}_{\text{cost}}}{\text{CPT}(50\%) \cdot \text{GPT4}_{\text{cost}} + (100\% - \text{CPT}(50\%)) \cdot \text{Mixtral}_{\text{cost}}}$$

$$\text{SavingRatio}_{80} = \frac{20\% \cdot \text{Mixtral}_{\text{cost}} + 80\% \cdot \text{GPT4}_{\text{cost}}}{\text{CPT}(80\%) \cdot \text{GPT4}_{\text{cost}} + (100\% - \text{CPT}(80\%)) \cdot \text{Mixtral}_{\text{cost}}}$$

MTBench		
Method	**Saving Ratio at CPT(50%)**	**Saving Ratio at CPT(50%)**
Logistic Regression	2.584	1.480
XGBoost	2.230	1.976
KNN	2.293	1.514
KNN+XGBoost	**2.759**	**2.006**

Table 5.2.1: Cost saving ratios on MTBench

GSM8K		
Method	**Saving Ratio at CPT(50%)**	**Saving Ratio at CPT(50%)**
Logistic Regression	1.413	1.199
XGBoost	**1.452**	**1.278**
KNN	1.186	1.048
KNN+XGBoost	1.166	1.151

Table 5.2.2: Cost saving ratios on GSM8K

MMLU		
Method	**Saving Ratio at CPT(50%)**	**Saving Ratio at CPT(80%)**
Logistic Regression	1.130	1.074
XGBoost	1.166	1.038
KNN	1.138	1.003
KNN+XGBoost	**1.201**	**1.077**

Table 5.2.3: Cost saving ratios on MMLU

The results from Tables 5.2.1, 5.2.2 and 5.2.3 demonstrate that different routing methods can lead to notable cost reductions across multiple benchmarks while maintaining quality. On MT Bench, the best-performing method (KNN+XGBoost) achieves up to 2.76× savings at CPT(50%) and 2.01× at CPT(80%), highlighting its efficiency in balancing performance and cost. For MMLU, improvements are more modest, with KNN+XGBoost again performing best, reaching 1.20× at CPT(50%) and 1.08× at CPT(80%). On GSM8K, XGBoost achieves the highest cost savings, with 1.45× at CPT(50%) and 1.28× at CPT(80%). These findings indicate that while the extent of cost savings varies across datasets, hybrid and boosted methods consistently outperform standalone approaches, demonstrating the effectiveness of routing strategies for reducing costs without significantly compromising response quality.

5.3 Objective Evaluation

Objective 1:

To design and evaluate four routing methods capable of efficiently selecting the most suitable large language model (LLM) for each query across diverse domains.

The methods were highly successful on the conversational MT-Bench benchmark, where the hybrid KNN+XGBoost model achieved a significant 41.80% performance improvement over a random baseline. However, efficiency dropped on specialized domains. On the GSM8K mathematical reasoning benchmark, most methods struggled, with only the standalone XGBoost model providing a modest 16.50% gain, highlighting the inadequacy of semantic similarity for such tasks. This challenge was most evident on the highly diverse MMLU benchmark, where nearly all models failed to provide any meaningful improvement. Strikingly, only the synergistic KNN+XGBoost model succeeded in this complex environment with an 11.4% improvement, proving that while the objective was technically met, practical effectiveness is limited and requires advanced architectures for broad, multi-domain applications. Therefore, objective 1 is considered **partially met**, while most routing methods provided some efficiency improvements, there were critical exceptions, most notably the standalone XGBoost model performing worse than random on MMLU and the near-total failure of the KNN router on the GSM8K benchmark.

Objective 2:

To analyse the trade-offs between accuracy, and cost in LLM routing, and to demonstrate that the proposed approach achieves significant cost savings on popular benchmarks with minimal impact on response quality.

Domain (Benchmark)	**Best Performing Router for Cost**	**Impact on Response Quality**	**Evaluation Summary**
MT-Bench	KNN+XGBoost *Up to 2.76× savings at CPT(50%) and 2.01× savings at CPT(80%).*	Minimal impact. The model achieved the highest overall routing efficiency (APGR 0.709), indicating high-quality responses were maintained.	**Achieved.** The router demonstrated a highly successful trade-off, delivering substantial cost reductions while simultaneously improving the quality of routing decisions over the baseline.
GSM8K	XGBoost *1.45× savings at CPT(50%) and 1.28× savings at CPT(80%).*	Minimal impact. While overall APGR improvement was modest, the CPT(80%) score of 62.40% surpassed the result from the	**Achieved.** The objective was met, with the router providing clear cost savings. A key achievement was outperforming a benchmark paper on a

		RouteLLM paper, showing it is highly adept at identifying the most difficult prompts needing the strong LLM.	critical efficiency metric, confirming a positive accuracy-to-cost trade-off.
MMLU	KNN+XGBoost *1.20× savings at CPT(50%) and 1.08× savings at CPT(80%).*	Minimal impact. Though cost savings were modest, this was the only router to achieve a meaningful performance gain (+11.4%), preventing the quality degradation seen in other models.	**Barely Achieved.** While cost savings were the lowest across the benchmarks, the router successfully demonstrated a positive trade-off by being the only method to improve response quality, thus preventing the financial waste of using ineffective models.

Table 5.3.1 Evaluation for objective 2

Based on the evaluation on Table 5.3.1, the objective to analyze the trade-offs between accuracy and cost was successfully achieved. The designed routing methods demonstrated significant cost savings across all benchmarks while having a minimal, and often positive, impact on response quality. On the conversational MT-Bench, the KNN+XGBoost router exemplified a highly successful trade-off, delivering substantial cost reductions of up to 2.76× while simultaneously achieving the highest routing efficiency. Similarly, on the more specialized GSM8K benchmark, the XGBoost router provided clear cost savings of up to 1.45× and confirmed a positive accuracy and cost balance by outperforming a key metric from the RouteLLM paper. Even in the most challenging MMLU domain, where savings were more modest at 1.20×, the KNN+XGBoost router proved its value by being the only method to prevent quality degradation, thus successfully demonstrating a positive trade-off by avoiding the financial waste associated with ineffective models.

CHAPTER 6 – Conclusion and Recommendations

6.1 Conclusion

The research successfully demonstrated that introducing a routing layer is a highly effective strategy for achieving significant cost savings with minimal impact on response quality. The findings confirmed that across conversational, mathematical, and multi-domain benchmarks, the proposed methods could reduce operational expenses by up to 2.76 times. A key achievement was the XGBoost router on the GSM8K benchmark, which not only provided clear cost benefits but also surpassed a performance metric (CPT(80%) by 0.86%) of their best tested model in the RouteLLM paper on a critical efficiency metric, validating the positive accuracy to cost trade-off. This research serves as a successful proof-of-concept, validating the core premise that an intelligent routing layer can deliver substantial financial and computational efficiencies. It affirms that the trade-off between cost and quality can be effectively managed without significant performance degradation and evaluated using the methodology in this experiment.

6.2 Limitations

While this research demonstrated some positive results in experimental settings, several limitations must be acknowledged. While this research successfully demonstrates the viability and cost-saving potential of an LLM routing layer, several limitations must be acknowledged. A primary limitation stems from the training data, which was labeled using an "LLM-as-a-judge" methodology. While scalable and efficient, this approach may not perfectly capture the nuances of human satisfaction and could introduce biases from the judge model itself. A router trained on these labels might optimize for what a powerful LLM deems a good response, rather than what an end-user truly finds helpful and satisfactory.

Furthermore, the positive results observed in this experimental setting, which relied on established benchmarks, may not directly translate to real-world applications. The distribution and nature of live user queries can differ substantially from the structured tasks found in benchmarks. The performance of the routers could therefore be impacted by the lack of a large-scale, diverse training dataset that mirrors the complexity of a live production environment.

6.3 Recommendations for Future Work

The findings and limitations of this research project open up several promising avenues for future work. A crucial next step would be to retrain and evaluate the routing models using a larger dataset labeled by human evaluators. This would create a more robust system aligned with genuine user satisfaction and provide a clearer picture of the routers' practical effectiveness.

Another significant direction is to extend the current two-model (strong vs. weak) framework to a more complex, multi-model environment. A system capable of routing queries across a wider spectrum of LLMs with varying capabilities, latencies, and costs would allow for even more granular and dynamic optimization.

REFERENCES

[1] "Introducing GPT-5." Accessed: Sept. 09, 2025. [Online]. Available: https://openai.com/index/introducing-gpt-5/

[2] G. Stefanelli, "ChatGPT-5 and user complaints: missing older models, colder replies, workflow issues, and more," Data Studios ·Exafin. Accessed: Sept. 09, 2025. [Online]. Available: https://www.datastudios.org/post/chatgpt-5-and-user-complaints-missing-older-models-colder-replies-workflow-issues-and-more

[3] S. B. Olam, "ChatGPT 5 Weaknesses Revealed," Olam News. Accessed: Sept. 09, 2025. [Online]. Available: https://www.olamnews.com/technology/ai/51/chatgpt5-weaknesses/

[4] A. Heath, "ChatGPT won't remove old models without warning after GPT-5 backlash," The Verge. Accessed: Sept. 09, 2025. [Online]. Available: https://www.theverge.com/openai/758537/chatgpt-4o-gpt-5-model-backlash-replacement

[5] I. Ong *et al.*, "RouteLLM: Learning to Route LLMs with Preference Data," Feb. 23, 2025, *arXiv*: arXiv:2406.18665. doi: 10.48550/arXiv.2406.18665.

[6] Z. Zhao, S. Jin, and Z. M. Mao, "Eagle: Efficient Training-Free Router for Multi-LLM Inference," Oct. 29, 2024, *arXiv*: arXiv:2409.15518. doi: 10.48550/arXiv.2409.15518.

[7] D. Stripelis *et al.*, "TensorOpera Router: A Multi-Model Router for Efficient LLM Inference," Oct. 23, 2024, *arXiv*: arXiv:2408.12320. doi: 10.48550/arXiv.2408.12320.

[8] R. Zhuang, T. Wu, Z. Wen, A. Li, J. Jiao, and K. Ramchandran, "EmbedLLM: Learning Compact Representations of Large Language Models," Oct. 16, 2024, *arXiv*: arXiv:2410.02223. doi: 10.48550/arXiv.2410.02223.

[9] Q. H. Nguyen *et al.*, "MetaLLM: A High-performant and Cost-efficient Dynamic Framework for Wrapping LLMs," Apr. 22, 2025, *arXiv*: arXiv:2407.10834. doi: 10.48550/arXiv.2407.10834.

[10] "Our next-generation model: Gemini 1.5," Google. Accessed: Apr. 22, 2025. [Online]. Available: https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/

[11] "RouteLLM: An Open-Source Framework for Cost-Effective LLM Routing | LMSYS Org." Accessed: Sept. 10, 2025. [Online]. Available: https://lmsys.org/blog/2024-07-01-routellm

[12] D. Hendrycks *et al.*, "Measuring Massive Multitask Language Understanding," Jan. 12, 2021, *arXiv*: arXiv:2009.03300. doi: 10.48550/arXiv.2009.03300.

[13] J. Chen *et al.*, "MTBench: A Multimodal Time Series Benchmark for Temporal Reasoning and Question Answering," Mar. 21, 2025, *arXiv*: arXiv:2503.16858. doi: 10.48550/arXiv.2503.16858.

[14] "FIGURE 2. Architecture of DistilRoBERTa.," ResearchGate. Accessed: Sept. 15, 2025. [Online]. Available: https://www.researchgate.net/figure/Architecture-of-DistilRoBERTa_fig1_359676326

[15] "sentence-transformers/all-distilroberta-v1 · Hugging Face." Accessed: Sept. 15, 2025. [Online]. Available: https://huggingface.co/sentence-transformers/all-distilroberta-v1

[16] R. Yassminh, “XGBoost with a Simple Example,” Medium. Accessed: Sept. 07, 2025. [Online]. Available: https://medium.com/@ryassminh/xgboost-with-a-simple-example-92d5d91789e2

[17] “routellm/gpt4_dataset · Datasets at Hugging Face.” Accessed: Sept. 07, 2025. [Online]. Available: https://huggingface.co/datasets/routellm/gpt4_dataset/viewer/default/train?views%5B%5D=train

[18] “Building an LLM Router for High-Quality and Cost-Effective Responses,” Anyscale. Accessed: Sept. 07, 2025. [Online]. Available: https://www.anyscale.com/blog/building-an-llm-router-for-high-quality-and-cost-effective-responses

[19] “GSM8K Benchmark — Klu.” Accessed: Apr. 11, 2025. [Online]. Available: https://klu.ai/glossary/GSM8K-eval

[20] “gpt-4-1106-preview by OpenAI on the AI Playground.” Accessed: Sept. 14, 2025. [Online]. Available: https://ai-sdk.dev/playground/openai:gpt-4-1106-preview

APPENDIX

Poster

UNIVERSITI TUNKU ABDUL RAHMAN

EFFICIENT SIMILARITY INDEX BASED TASK ROUTING FOR LLM

CANDIDATE
Liew Zheng Xian 21ACB05330

SUPERVISOR
Dr Aun Yichiet

FACULTY OF INFORMATION AND COMMUNICATION TECHNOLOGY

INTRODUCTION

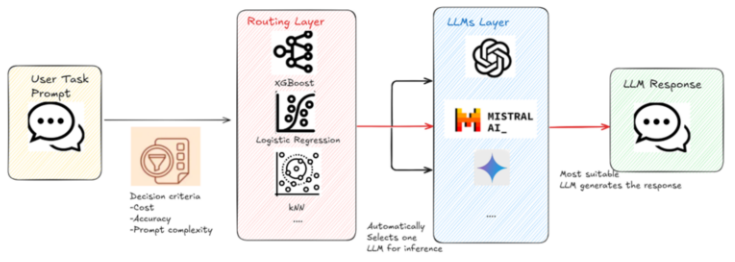
Large Language Models (LLMs) represent a significant innovation within the field of Generative AI (GenAI), yet their practical deployment is hampered by a critical trade-off between the high performance of large models and the cost-efficiency of smaller ones. This research addresses this challenge by designing and evaluating an intelligent LLM routing system to optimize both cost and performance.

OBJECTIVES

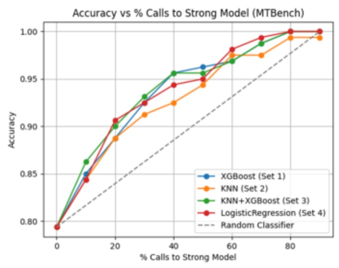
1.To design and evaluate four routing methods capable of efficiently selecting the most suitable large language model (LLM) for each query across diverse domains.

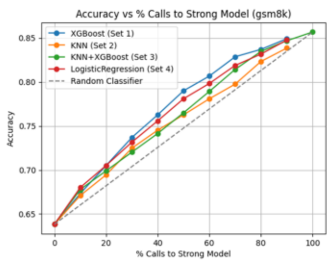
2.To analyse the trade-offs between accuracy, and cost in LLM routing, and to demonstrate that the proposed approach achieves significant cost savings on popular benchmarks with minimal impact on response quality.

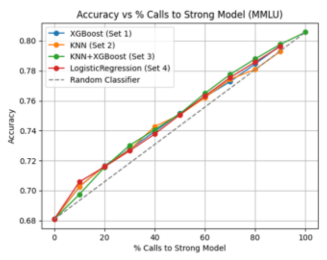
SYSTEM MODEL



RESULTS







CONCLUSION

This research successfully demonstrates that an intelligent routing layer is a highly effective strategy for managing the trade-off between LLM cost and quality. The system achieved up to 2.76x in cost savings across diverse benchmarks and, as a key achievement, surpassed a critical efficiency metric from the RouteLLM paper. This serves as a successful proof-of-concept, validating that routing can deliver substantial financial efficiencies without significant performance degradation.